Lucas Rinaldi

Sistema de Atendimento ao Consumidor Utilizando a Tecnologia WebRTC

Florianópolis

2018

Lucas Rinaldi

Sistema de Atendimento ao Consumidor Utilizando a Tecnologia WebRTC

Trabalho apresentado ao Curso de Sistemas de Informação da Universidade Federal de Santa Catarina como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Universidade Federal de Santa Catarina – UFSC

Centro Tecnológico – CTC

Departamento de Informática e Estatística – INE

Orientador: Prof. Frank Siqueira

Florianópolis

2018

RESUMO

O objetivo desse trabalho é melhorar a forma como *e-commerce*, suporte técnico e serviços online fornecem atendimento ao cliente implementando um sistema de comunicação por vídeo que possa ser incluído em *websites* de forma simples e rápida. Para a transmissão de vídeos em tempo real será utilizada a tecnologia WebRTC, que apesar de ser relativamente nova, está expandindo horizontes na questão de transmissão de vídeos. Essa tecnologia torna desnecessário o uso de plugins e bibliotecas externas para comunicação através de áudio e vídeo no *browser*, simplificando o suporte técnico.

Palavras-chaves: sistemas web, webrtc, redes ponto-a-ponto, videoconferência, suporte

ABSTRACT

The goal of this paper is to improve the way e-commerce, technical support and online services provide customer support by implementing a communication system through video that can easily be included in companies' websites. Realtime video communication will be done using the WebRTC technology, which is relatively new but is expanding in a fast pace in video streaming. One if its advantages is that it makes unnecessary to use plugins or external libraries to transmit audio and video inside the browser, simplifying the customer support.

Keywords: web systems, webrtc, peer-to-peer network, videoconference, support

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CGI	Common Gateway Interface
CSRF	Cross-site request forgery
DOM	Document Object Model
HLS	HTTP Live Streaming
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICE	Interactive Connectivity Establishment
IETF	Internet Engineering Task Force
IP	Internet Protocol
NAT	Network Address Translator
PaaS	Platform as a Service
P2P	Peer-to-Peer
REST	Representational State Transfer
RF	Requisitos Funcionais
RNF	Requisitos Não-Funcionais
RPTM	Real-time Messaging Protocol
SAC	Serviço de Atendimento ao Consumidor
SOAP	Simple Object Access Protocol
SPA	Single-Page Application
SQL	Structured Query Language
SSL	Secure Sockets Layer

STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
TI	Tecnologia da Informação
TLS	Transport Layer Security
TURN	Traversal Using Relays around NAT
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WSDL	Web Services Description Language
W3C	World Wide Web Consortium

1 INTRODUÇÃO

Com a crescente competitividade e concorrência de mercado, uma das maneiras de um negócio se destacar e conseguir sobreviver nessa realidade é investir no relacionamento com o cliente. Esse investimento busca melhorar o grau de satisfação do cliente, que possui um papel muito importante na longevidade do negócio.

Podemos identificar alguns problemas que afetam empresas no mercado atual em relação a este assunto:

- Empresas estão falhando em conhecer com precisão o perfil de seus consumidores;
- Há grande preocupação em adquirir novos clientes, mas não em retê-los. De acordo com [Farris et al. 2015], o preço para conseguir um novo cliente pode ser até 14 vezes maior que nutrir e manter um que já está na base;
- A tecnologia no atendimento ao cliente resultou em impessoalidade, com cada vez mais robôs atendentes no lugar de pessoas.

Uma das maneiras utilizadas para realizar a comunicação com o cliente é através do Sistema de Atendimento ao Consumidor - mais conhecido como SAC - de cada empresa. O SAC pode ser visto como uma ferramenta para solucionar a grande demanda de dúvidas, solicitações e reclamações de sua base de clientes, consistindo em uma das possíveis estratégias para desenvolver um bom relacionamento com a base de clientes de uma empresa.

Dados trazidos por [Dixon, Freeman e Toman 2010] mostram que 48% dos consumidores que tiveram experiências negativas com o SAC contam a dez ou mais pessoas; enquanto somente 23% com experiências positivas contam a dez ou mais pessoas. Passamos da fase em que o atendimento ao cliente é um mero suporte, e ser eficiente nesse processo pode garantir a longevidade do seu negócio.

Alguns problemas encontrados em sistemas de atendimento ao consumidor são:

- Um novo perfil de consumidor surgiu no mercado (os da Geração Y, ou Millennials). Esse perfil é mais fluído e versátil, exigindo inovação dos profissionais de atendimento [Salesforce 2015];
- Burocracia gerada por processos e tecnologias engessadas e obsoletas criam fluxos confusos e exaustivos para os consumidores, deixando o SAC no passado e por consequência ineficiente;

- As empresas não estão nos mesmos canais que os clientes e não conseguem acompanhar as mudanças rápidas que a Internet, as redes sociais e as novas tecnologias provocam nos consumidores todos os dias.

- O principal meio de atendimento ainda é via telefone (de acordo com [ConsumidorModerno 2015]). O consumidor atual não se encaixa mais nesse perfil. Dados mostram que 58% dos brasileiros não estão dispostos a esperar mais que 5 minutos por uma resposta ao telefone de um SAC.

Com base nas necessidades do mercado de um melhor relacionamento com o cliente, nas deficiências dos atuais sistemas de atendimento e nas limitações técnicas que alguns consumidores enfrentam, decidimos desenvolver um Sistema de Atendimento ao Consumidor que funcione através de transmissão de vídeo utilizando a tecnologia WebRTC.

1.1 MOTIVAÇÃO

Observa-se a existência de diversas aplicações *web* voltadas para atendimento ao consumidor, com foco em soluções de *tickets*, análise e categorização dos mesmos.

Entre elas é possível citar Zendesk[1] e Desk.com[2] como estado da arte desse segmento. O mesmo acontece no segmento de programas voltados para conversas através de vídeo. No estado da arte dessa categoria podemos citar appear.in[3], disponível como aplicação web; Hangouts[4], com versão Web e aplicativo móvel; e o Skype[5], disponível para Web, *desktop* e dispositivos móveis.

Entretanto a correlação dos dois segmentos é pouco explorada, tanto no âmbito da Web, como com programas *desktop*, ou seja, sem a necessidade de instalação de programas e *plugins*. Para o usuário, a facilidade de acesso e a independência do sistema operacional são vantagens de uma aplicação web em relação à sua versão para uma plataforma específica.

Essa foi a principal motivação para este projeto: o desenvolvimento de uma aplicação web que faça a gerência de chamadas em vídeo e seja de fácil acesso para qualquer tipo de dispositivo. As chamadas iniciam-se por clientes que necessitam de assistência para o produto oferecido pela empresa, e são respondidas por atendentes designados para esse tipo de trabalho.

[1] Disponível em: <https://www.zendesk.com/>. Acesso em 30 jun. 2017.
[2] Disponível em: <https://www.desk.com/>. Acesso em 30 jun. 2017.
[3] Disponível em: <https://www.appear.in/>. Acesso em 30 jun. 2017.
[4] Disponível em: <https://hangouts.google.com/>. Acesso em 30 jun. 2017.
[5] Disponível em: <https://www.skype.com/>. Acesso em 30 jun. 2017.

1.2 OBJETIVO

1.2.1 Objetivo Geral

Desenvolver um Sistema de Atendimento ao Consumidor que funcione através de transmissão de vídeo. O sistema deverá dispor de uma área administrativa, onde um profissional responsável por prestar atendimento gerenciará seus chamados.

1.2.2 Objetivos Específicos

Os seguintes objetivos específicos são almejados por este trabalho:

- Estudar as soluções existentes para atendimento ao consumidor por chamadas de vídeo e identificar os problemas existentes;
- Desenvolver controle de acesso por parte dos atendentes.
- Desenvolver uma API baseada no modelo *RESTful* para cadastro de usuários e chamadas através de um servidor HTTP.
- Desenvolver um módulo que oferece conexões através de *WebSockets* para:
  - Receber requisições de chamadas em vídeo *realtime*.
  - Realizar o *handshake* necessário para conexão *peer-to-peer* entre navegadores.
- Implementar interface de aplicação web que suporte atualização em tempo real de componentes sem que seja necessário a atualização da página.

1.3 JUSTIFICATIVA

O programa desenvolvido será utilizado para resolver tanto problemas de negócio como problemas técnicos. Auxiliará no gerenciamento de chamadas de vídeo entre consumidores e profissionais de suporte/atendimento. Irá segurar chamadas, atendê-las e encerrá-las, mostrando dados e a localidade do cliente.

O modelo de atendimento por vídeo faz as empresas modernas retomarem a proximidade com os clientes. A tecnologia pode ser utilizada para sermos mais pessoais e alcançarmos uma gama maior de perfis de consumidor, por exemplo: o tipo de consumidor que está acostumado a interagir com o vídeo - meio de comunicação que se tornou uma das principais ferramentas de ajuda e comunicação na nossa sociedade atual.

De acordo com [Chazal 2015], o vídeo é o canal de crescimento mais rápido. O número de chamadas de vídeo subiu de 600 milhões em 2010 para 30 bilhões em 2015.

Enquanto isso, 2 bilhões de minutos de chamadas via Skype são feitos todos os dias e 36% dos consumidores já expressam o desejo de terem suporte por meio de vídeo.

No entanto, a transmissão por vídeo é uma tecnologia relativamente nova em sistemas de atendimento, e somente 0,2% das empresas prestam atendimento via vídeo (de acordo com a [NewVoiceMedia 2014]). Empresas de consultoria, da área da saúde, *call* centers e advogados, por exemplo, já adotaram esse modelo.

No âmbito jurídico, o vídeo já está sendo utilizado para colher depoimentos em processos. Para áreas de recursos humanos, diversas empresas utilizam a chamada por vídeo para realizar entrevistas com candidatos de outra localidade, sendo que aproximadamente 60% do gerentes usam esse recurso ([Huspeni 2013]).

Já a transmissão de vídeo através de *smartphones* e computadores pessoais não é um assunto tão recente. Utilizamos programas como Skype e *plug-ins* de terceiros (ex.: Flash) para realizar chamadas entre dispositivos. Isso gera um *overhead* para o consumidor, que fica a mercê da empresa para saber qual ferramenta utilizar para a chamada. Além disso, o usuário pode não ter em seu *smartphone* o aplicativo ou *plug-in* pré-instalado, e pode não dispor de espaço de armazenamento em seu dispositivo para realizar a instalação do software. O sistema desenvolvido precisará somente de um navegador (Google Chrome, Firefox, etc) em ambos nodos da conversação, ou seja, será suportado pela maioria dos *smartphones*, computadores pessoais e *tablets*. Assim, evitamos a necessidade de o usuário instalar programas adicionais em seu dispositivo.

1.4 METODOLOGIA

Visando o sucesso deste projeto, foi adotada uma metodologia de trabalho que, no primeiro momento, consiste em reuniões periódicas com o orientador do trabalho de conclusão a fim de levantarmos requisitos e necessidades existentes do projeto.

Na segunda etapa foram realizadas análises de relatórios, artigos e pesquisas sobre dificuldades que consumidores têm em relação ao serviços de atendimento ao consumidor de uma grande parcela das empresas brasileiras. Foi observado que grande parte dos desejos e reclamações dos clientes dizem respeito à SACs com possibilidade de chamada em vídeo.

Com essa informação em mãos, uma pesquisa foi feita para encontrar as dificuldades que as empresas têm em fornecer um atendimento em vídeo de qualidade e fácil acesso. Análise feita em conjunto com sistemas utilizados para chamadas de videoconferência.

Para justificar a implementação do sistema foram levantados requisitos unindo as dificuldades tanto dos consumidores quanto das empresas em realizar chamados de assistência técnica bem sucedidos através da transmissão de vídeo.

A solução encontrada diz respeito a um sistema que:

- Seja multi-dispositivo, ou seja, que funcione tanto em computadores pessoais como em dispositivos móveis;
- Multi-plataforma, isto é, que esteja disponível em diversos sistemas operacionais, tanto para computadores (*Windows*, *Linux*, *Mac OS*) quanto para dispositivos móveis (*iOS*, *Android*);
- Fácil ou nenhuma instalação, sem necessidade de instalação de aplicativos proprietários;
- Possua gerenciamento de chamadas.

Construído o modelo e as ideias da aplicação, o projeto de implementação foi desenvolvido. O projeto constitui-se de análise de requisitos, descrição de tecnologias e arquitetura, junto das suas funcionalidades.

Com o projeto de implementação em mãos, o desenvolvimento do sistema será feito através de uma plataforma chamada *NodeJS*, utilizando dois *frameworks*: *Express.js* e *Socket.IO*, o primeiro responsável pela API e o segundo para conexão através de *WebSockets*. O servidor que fornece conexão por *sockets* será responsável por realizar o estabelecimento da conexão entre dois nodos da rede, gerenciar chamadas e a API por guardar informações importantes sobre as mesmas.

Após a conexão entre duas partes a transmissão de mídia (principalmente vídeo) será realizada por uma tecnologia moderna chamada *WebRTC*, especializada em transmissão de mídia em tempo real.

Por fim, a publicação de todo sistema será feito através do *Heroku*, uma empresa de *PaaS* que fornece toda assistência para fornecer as aplicações online de forma simples e eficiente.

1.5 ORGANIZAÇÃO DO TEXTO

Conhecendo os objetivos, metodologia e a motivação por trás deste projeto, os próximos capítulos dão início às etapas de trabalho propostas.

Apresenta-se no capítulo seguinte a fundamentação teórica: descrições sobre os conceitos teóricos que cercam o sistema a ser implementado, descrição sobre conceitos utilizados por outras aplicações do mesmo ramo e a evolução da tecnologia que foi usada como base na construção do projeto (WebRTC);

O terceiro capítulo contém uma análise das ferramentas existentes tanto na categoria de atendimento ao consumidor, como na de videoconferência, e uma breve discussão sobre os pontos de melhoria em comparação com este projeto.

No quarto capítulo é descrita a analise de requisitos do software, ou seja, as funcionalidades que serão implementadas no mesmo. Junto dos requisitos serão demonstradas a modelagem e a arquitetura do projeto.

A implementação do projeto encontra-se no quinto capítulo, no qual serão descritas a análise de requisitos, as decisões e descrições de tecnologias e ferramentas utilizadas, a arquitetura escolhida e as funcionalidades do sistema.

2 FUNDAMENTAÇÃO TEÓRICA

Para que seja possível uma completa compreensão do trabalho e de como usar o sistema por qualquer leitor, independentemente do nível de conhecimento sobre os principais tópicos do trabalho, é feita uma breve definição e elucidação destes tópicos nesse capítulo. Todos os conceitos descritos aqui assumem um prévio conhecimento do que é a Internet e a *World Wide Web*.

2.1 ARQUITETURA PEER-TO-PEER

Estilo de arquitetura que possui somente um tipo de entidade, essa definida de *Servent* por [Schollmeier 2001, Sec. 2]. *Servent* vem da junção das palavras *Server* e *Client*, indicando que cada *peer* age como um cliente e um servidor. Dito isso, essa arquitetura permite que nodos heterogêneos e distintos possam comunicar-se entre si sem a necessidade de um servidor [Ripeanu 2001].

Podemos ver pela Figura 1 que a grande diferença dessa arquitetura para a cliente-servidor, é que a última possuí uma entidade de grande processamento, que possui todos os recursos e todos os nodos se conectam a ele, que é o Servidor.

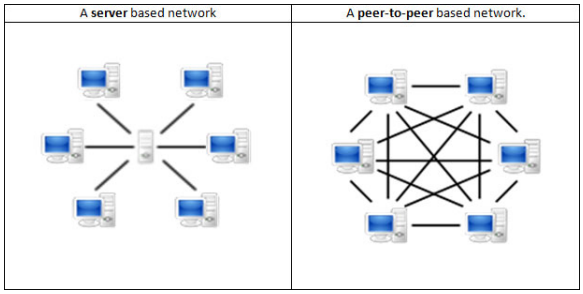

Figura 1 – Arquiteturas de rede

Podemos dizer então que uma arquitetura distribuída pode ser chamada de *peer-to-peer* se cada um dos seus participantes compartilhar parte dos seus recursos, seja processamento, armazenamento, banda larga. O que define quais recursos são, é o tipo de rede que está utilizando a arquitetura.

2.2 ARQUITETURA REST

Representational State Transfer (REST) é um estilo de arquitetura dedicado a sistemas distribuídos, que utiliza *RESTful web services* (serviços Web compatíveis com

REST) para facilitar a interoperabilidade de sistemas dentro de uma rede, como a Internet. O termo surgiu pela primeira vez no ano 2000 em uma dissertação de Roy Fielding, um dos principais autores da especificação HTTP.

De acordo com [Fielding 2000] a lógica por trás de uma arquitetura para *Web* é descrita por um conjunto de restrições aplicadas aos elementos da arquitetura.

REST, utiliza padrões e restrições para definir como recursos da Internet devem ser definidos e utilizados. Algumas das restrições são:

- Utilizar arquitetura cliente-servidor.
- Toda operação é *stateless*, ou seja, não guarda estado; cada requisição ao servidor deve conter toda informação necessária para realizar a operação.
- O sistema deve implementar uma camada de *cache* para melhorar a eficiência na rede (devido à natureza *stateless*).

Interoperabilidade entre sistemas, significa acima de tudo a possibilidade de troca de informações dentro da arquitetura. Observamos na tabela 1 a definição de elementos de dados que especificam o tipo de informação disponível e como chegar até ela.

Elemento de dados	**Exemplos da web**
resource	Mapeamento conceitual de uma referência em hipertexto
resource identifier	URL, URN, URI
representation	Documento HTML, imagem JPEG
representation metadata	Tipo de mídia, última data de modificação
resource metadata	Fonte do link
control data	if-modified-since, cache control (cabeçalhos HTTP)

Tabela 1 – Elementos de dados REST

O conceito chave de informação no REST é *resource*. Esse pode ser um texto, uma imagem, um serviço, dentre outros. Para identificar um recurso específico no servidor utiliza-se identificadores de recursos (*resource identifiers*), usualmente conhecidos como *link* ou URL.

Ao acessar um identificador recebemos uma representação (*representation*) que remete ao estado (*state*) do recurso em determinado momento no tempo.

REST geralmente é implementado por servidores sobre o protocolo de comunicação HTTP utilizando os mesmos verbos disponíveis (GET, POST, PUT, DELETE) no protocolo. No servidor é definido um conjunto de "operações sem estado"e através dessas operações os servidores tem acesso aos chamados Web resources.

Existem outros serviços da Web que fornecem interoperabilidade e a noção de recursos, porém com suas próprias características, como por exemplo, WSDL e SOAP.

Esse estilo de arquitetura foi escolhido para o trabalho por ser o mais difundido na comunidade hoje em dia. No sistema de atendimento ao consumidor será utilizada a arquitetura REST em um servidor API com o objetivo de salvar as interações dos usuários com o nosso sistema (chamadas de vídeo, requisições de atendimento, dentre outras).

2.3 PROTOCOLO HTTP

HyperText Transfer Protocol (em português protocolo de transferência de hipertexto) é um protocolo utilizado em nível de aplicação em modelos como o TCP/IP. Serve principalmente para transferir informações em sistemas distribuídos como a *World Wide Web*, provavelmente o mais popular.

Dentro de um sistema massivo de informação como a Internet o protocolo HTTP funciona através de operações requisição-resposta sem estado. De acordo com [Fielding et al. 1999, Sec. 4] essas operações são mensagens de texto que obedecem a um padrão definido pela especificação HTTP.

Existem duas entidades principais em uma mensagem HTTP. O cliente que envia uma mensagem requisitando um recurso, e o servidor que a processa e responde de acordo com os dados da requisição.

A mensagem de requisição deve especificar um método dentre um conjunto definido pelo protocolo de transferência. Conforme [Fielding et al. 1999, Sec. 9], cada método do protocolo determina o tipo de operação feita no servidor e o resultado a ser esperado. A Tabela 2 descreve os principais métodos definidos na especificação do protocolo.

Método	**Descrição**
GET	Retorna dados de um recurso específico.
DELETE	Remove um recurso específico.
POST	Modifica ou altera um recurso. Não cria.
PUT	Cria ou sobrescreve um recurso.
OPTIONS	Lista as opções de comunicação com o recurso.

Tabela 2 – Métodos HTTP

A responsabilidade do servidor é responder com o código de estado e o recurso requerido, se existente. Observamos em [Fielding et al. 1999, Sec. 10] que o código de status faz parte de um conjunto de números que refletem o resultado da requisição. Consulte a Tabela 3, na qual encontram-se os códigos mais usados e o seu significado.

Fundamental em toda aplicação que deseja buscar ou enviar informações através

Código de estado	Significado
200	Requisição bem sucedida.
301	Recurso movido permanentemente.
404	Recurso não encontrado.
500	Erro no servidor.

Tabela 3 – Códigos HTTP

da Internet, pode ser usado como base para outros tipos de protocolo, como o protocolo *WebSocket* que será abordado no próximo capítulo, ou também para arquiteturas como REST, abordada na seção anterior.

2.4 PROTOCOLO WEBSOCKET

Com o avanço da Internet, surgiram novos tipos de aplicações como alternativa ao paradigma usual de requisição-resposta. A Internet tornou-se uma plataforma onde envia-se todo tipo de dado para comunicação entre computadores. O surgimento de jogos online, programas de mensagem instantânea e aplicações em tempo real de modo geral, mostrou que é necessário avançar além da versão 1.0 do protocolo HTTP. Algumas estratégias foram criadas para mitigar esse problemas, uma delas chamada *long polling*. Nessa técnica após a primeira requisição o servidor mantém a conexão TCP aberta até ter novos dados para enviar ao cliente. Quando o cliente recebe os novos dados, automaticamente realiza uma nova solicitação. Usar esse tipo de técnica traz algumas desvantagens, de acordo com [Fette e Melnikov 2011, Sec. 1]:

- Repetidas conexões TCP.
- Sobrecarga na conexão por ter que enviar cabeçalhos HTTP para cada mensagem trocada entre cliente e servidor.
- A nível de código é necessário implementar um gerenciador de requisições e respostas.

Como solução foi proposto o protocolo WebSocket, que utiliza somente uma conexão TCP bidirecional, sem sobrecarga de requisições, para comunicação entre as duas partes.

O protocolo WebSocket é uma camada desenvolvida para camada de transporte e é baseada no protocolo TCP, faz uso do protocolo IP para troca de mensagens. É baseada no TCP propositalmente para que seja compatível com servidores antigos que ainda não suportam o novo protocolo, e envolve duas partes, abertura de conexão e transferência de dados. A primeira é uma requisição HTTP feita pelo cliente indicando (através de cabeçalhos) que quer atualizar a conexão para WebSocket. Caso o servidor entenda

o protocolo, concordará em fazer a troca. Esse processo todo foi nomeado WebSocket *handshake*.

Com o *handshake* aceito, a conexão está estabelecida e agora existe um canal bidirecional entre cliente e servidor, pela qual cada parte pode enviar e receber dados a qualquer momento. Cada unidade de dados transferida é chamada de mensagem ou *frame* (a nível de cabeamento).

O protocolo é utilizado de forma subjacente na aplicação pelo *framework* Socket.IO (abordada no capítulo de projetos). Essa tecnologia facilita a implementação do servidor responsável pelo processo de *signaling* (em português, sinalização), que é similar ao *handshake* e necessário para realizar conexões ponto-a-ponto através do WebRTC.

2.5 WEBRTC

Desde a criação da Web, seu objetivo foi sempre o de permitir o acesso a informações de forma fácil e rápida. No primeiro momento textos com *hyperlinks* levavam a outros textos, e depois a imagens, áudios, vídeos, ou seja, todo tipo de mídia capaz de ser reproduzida digitalmente. Porém, todos representados estaticamente.

Depois da Google disponibilizar *codecs* de áudio e vídeo e outras técnicas utilizadas para comunicação em tempo real e criar grupo de pessoas responsáveis por padronizar e aprimorar as tecnologias criando especificações. Em 2011 a Ericsson surgiu com a primeira implementação do WebRTC ([Håkansson 2011]).

A tecnologia WebRTC é composta de um conjunto de especificações criadas por grupos de pessoas de instituições como W3C e IETF, além de empresas fabricantes de navegadores. Esses documentos têm como objetivo padronizar interfaces que serão responsáveis por realizar chamadas em tempo real, e os fabricantes de navegadores ficam com a responsabilidade de implementá-las em seus produtos. Com isso, busca-se permitir a comunicação entre usuários que utilizam diferentes *browsers*.

Atualmente as principais especificações envolvem três interfaces de programação de aplicações, que serão detalhadas nas próximas seções:

- MediaStream (ou getUserMedia) [Burnett et al. 2017, Sec. 4.2]
- RTCPeerConnection [Bergkvist et al. 2017, Sec. 4.4]
- RTCDataChannel [Bergkvist et al. 2017, Sec. 4.6]

2.5.1 MediaStream

Interface responsável por consumir fontes de áudio e vídeo em forma de fluxo de dados e controlar para onde esse fluxo vai ser direcionado – por exemplo, uma saída de áudio, elemento HTML, dentre outros. A especificação também define funções que fornecem acesso a dispositivos de mídia, como câmeras e microfones, de acordo com as permissões do usuário.

Com o acesso permitido, as informações chegam como *streams* (fluxos) de mídia e são conectadas ao *input* de um objeto MediaStream. Esse objeto possui múltiplas trilhas (*tracks*) e cada uma delas corresponde a um tipo diferente de mídia (vídeo de uma câmera, música de um mp3, etc.).

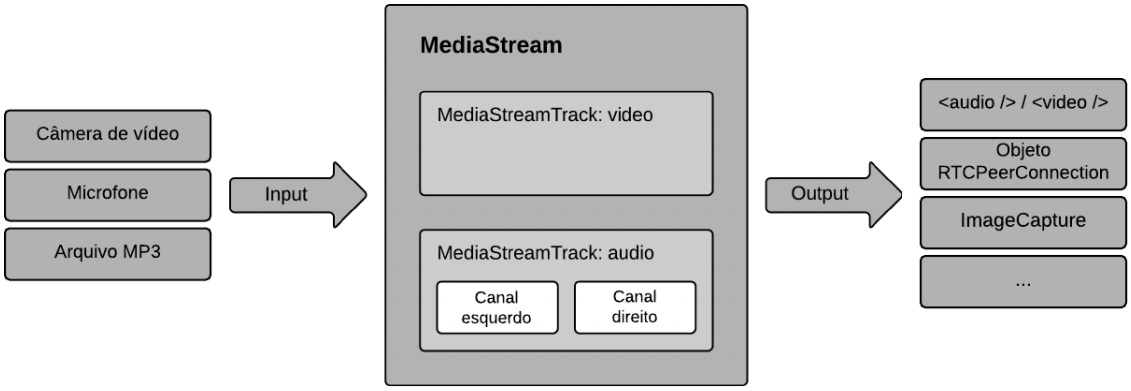

Figura 2 – Fluxo de dados usando MediaStream

Os chamados consumidores de MediaStream são objetos que conseguem ler o fluxo de dados do *output* dessa interface. A imagem 2 mostra exemplos de saídas existentes. A maior vantagem para o WebRTC é o fluxo poder ser enviado através de objetos RTCPeerConnection, sendo transportado como *streams*, permitindo assim chamadas de áudio e vídeo remotas. Os elementos HTML de áudio e vídeo são responsáveis pela reprodução de mídias no cliente e aceitam saídas MediaStream como entradas de dados.

2.5.2 RTCPeerConnection

De acordo com [Bergkvist et al. 2017, Sec. 4.1] é possível realizar conexões ponto-a-ponto entre diferentes computadores em uma rede através de dois objetos RTCPeerConnection. A conexão é realizada através de uma troca de mensagens padronizada por um protocolo de sinalização, similar ao *handshake* do TCP, porém no nível de aplicação.

Essa troca é chamada de *signaling* e não está definida na especificação, cabendo ao desenvolvedor escolher a maneira de realizá-la. Geralmente é utilizado um servidor que funciona através de WebSockets (no caso desse projeto), requisições HTTP, dentre outros mecanismos.

Em projetos WebRTC executados em uma única aba no navegador não é necessário o uso de um servidor, as informações estão no mesmo contexto de código e conseguimos realizar a conexão sem troca de mensagens através da Internet. Quando alteramos a aplicação para o âmbito da Web, funcionando em produção com peers remotos, basicamente precisamos de quatro funcionalidades no lado dos servidores:

- Tradução de endereço local para público;
- Descobrir endereço de IP público do próprio usuário (não do *peer* remoto);
- Retransmissão de dados através de servidor reserva caso a conexão ponto-a-ponto não funcione, ou não seja suportada;
- Sinalização do cliente enviada para o remoto indicando conexão P2P.

Todos esses pontos, exceto o da sinalização, são resolvidos por um protocolo chamado *Interactive Connectivity Establishment* (ICE), que geralmente é utilizado por aplicações WebRTC.

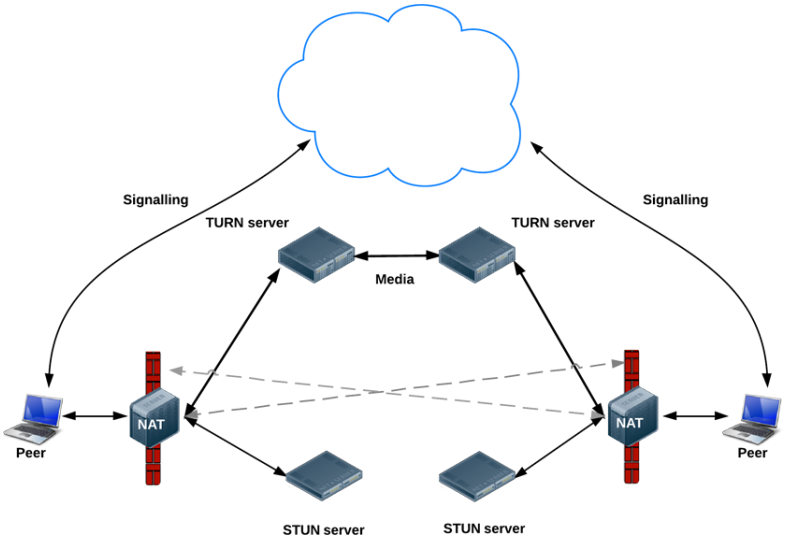

Figura 3 – Protocolo ICE, retirado de [Dutton 2013]

A figura 3 apresenta uma visão geral do processo de *signaling* e seus componentes principais, que são responsáveis cada um por uma funcionalidade citada acima:

- NAT - Responsável por gerar um endereço IP público através de uma tabela *hash* para o cliente;
- STUN - Servidor público ou privado, responde com endereço de IP gerado pelo NAT através de uma requisição feita pelo mesmo cliente;

- TURN - Servidor responsável por retransmitir os dados caso a conexão ponto-a-ponto não funcione.

Com a conexão *peer-to-peer* estabelecida, o servidor deixa de ser necessário e os navegadores estão aptos a transmitir informações em tempo real.

2.5.3 RTCDataChannel

Além de enviar áudio e vídeo, WebRTC tem a capacidade de se comunicar em tempo real por meio da interface RTCDataChannel usando diversos tipos de dados.

Essa funcionalidade foi implementada devido a certos casos de uso, como jogos online, transferência de arquivos, documentos de textos colaborativos (por exemplo, Google Docs), dentre outros.

O objetivo é atingir uma baixa latência aliada a uma alta capacidade de transmissão de dados, devido à redução de sobrecarga por *handshakes* TCP e cabeçalhos HTTP. É utilizado o mesmo tipo de conexão para fluxos de áudio e vídeo, através de um objeto RTCPeerConnection. Apesar de muito similar a conexões WebSockets, é mais rápida devido ao fato de estabelecer uma conexão direta de um navegador a outro.

WebRTC é a fundação do sistema de atendimento implementado nesse projeto. A capacidade de comunicação em tempo real, utilização diretamente do navegador e a não necessidade de instalar *plugins* de terceiros, tornaram a ferramenta ideal para solucionar os problemas apresentados no início do trabalho.

3 TRABALHOS RELACIONADOS

Nesta seção serão apresentados projetos relacionados ao tema do presente trabalho. Por ser um tema que agrupa dois assuntos, videoconferência e sistema de atendimento ao consumidor, e pela falta de softwares semelhantes, os dois assuntos serão analisados separadamente.

3.1 PROTOCOLOS PARA TRANSMISSÃO DE VÍDEO

3.1.1 Protocolo RTMP

O Protocolo RTMP (*Real-time Messaging Protocol*, [Parmar 2012]), desenvolvido pela Macromedia inicialmente como um produto proprietário e de código fechado, foi disponibilizado quando a Adobe adquiriu a empresa e liberou parte da especificação do protocolo

De acordo com sua especificação, o RMTP fornece um serviço bi-direcional de mensagens, tendo como função o *streaming* de áudio, vídeo e dados de alta performance através da Internet.

O protocolo funciona em cima de serviços de entrega de pacotes confiáveis como TCP, podendo assim ser utilizado sobre HTTP e HTTPS.

Conforme ilustrado na Figura 4, o RMTP é responsável por realizar a conexão entre um reprodutor Flash e um servidor de mídia Flash.

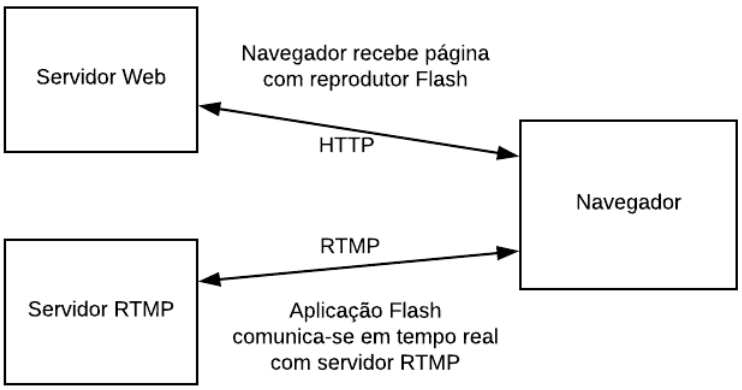

Figura 4 – Conexão entre reprodutor e servidor Flash através do RTMP

O reprodutor Flash foi inicialmente desenvolvido para reprodução de animações

vetoriais bidimensionais pela Macromedia, mas acabou por tornar-se uma boa escolha para *streaming* de mídia na Internet por sua capacidade de minimizar o tamanho do arquivo, economizando banda e diminuindo a latência.

Somente na versão Flash 10 o suporte a conexões P2P com RTMP foi disponibilizado.

Há cerca de uma década, o uso do RTMP era uma escolha mais segura no lugar do *WebRTC*, devido ao amplo suporte existente, resultante da presença de *plugins* Flash nos navegadores, além de reprodutores nativos. No entanto, esta situação mudou completamente nos últimos anos. Alguns pontos foram decisivos para a queda do Flash e do RTMP e a ascensão do *WebRTC*:

- O código fechado e proprietário do RTMP não agradava certas empresas. Adiciona-se a isso o fato de elas não poderem utilizá-lo sem um reprodutor Flash.
- *Plugins* de reprodutores Flash traziam problemas aos navegadores. Os fabricantes de *browsers* começaram a notar diversos gargalos de performance e segurança. Devido à obrigatoriedade do seu uso, surgiram novas frentes para criar tecnologias de *streaming*.

Passou-se a buscar uma tecnologia que não obrigasse o usuário a instalar *plugins* no navegador, ou seja, para a qual bastasse o suporte nativo existente.

Não muito tempo depois a Apple removeu o uso do reprodutor Flash nos seus navegadores (presentes em iPhones e iPads) fazendo com que a utilização do protocolo diminuísse. Em 2009 a Apple lançou o HLS (*HTTP Live Streaming*), que será descrito em seguida.

Praticamente ao mesmo tempo a iniciativa de desenvolvimento do HTML5 surgia. A especificação foi apoiada por Steve Jobs, que escreveu uma longa carta *Thoughts on Flash*[1] onde explica a razão de não colocar o reprodutor nos seus aparelhos.

Em questão de performance, as duas tecnologias têm suas vantagens e desvantagens. Foi devido ao modo de uso e a presença delas no dia-a-dia e ao crescente suporte dos navegadores ao *WebRTC* e ao HTML5, que RTMP tornou-se uma opção válida somente para casos específicos de uso.

3.1.2 Protocolo HLS

Protocolo de comunicação implementado pela Apple. Está presente em todos os seus aparelhos e softwares como QuickTime, Safari e iOS.

Funciona sem qualquer configuração adicional em servidores Web convencionais porque realiza todas as suas requisições através de conexões HTTP. Grande vantagem

[1] Disponível em: <https://www.apple.com/hotnews/thoughts-on-flash/>. Acesso em 20 mai. 2018.

pois é compatível com serviços dedicados a entrega de dados, as chamadas CDN (*Content Delivery Network*).

Sua especificação define uma arquitetura com três componentes principais[2]:

- Servidor, responsável por codificar e segmentar o arquivo de mídia.
- Distribuidor, um servidor HTTP que responde com a mídia e o arquivo de indexação necessário para reproduzi-la.
- Cliente, reprodutor de mídia com suporte a *streams*.

Popular pela qualidade de transmissão, adaptabilidade a diferentes conexões e disponibilidade de serviço. Características existentes devido ao segmentador, componente responsável pela separação da mídia, após a codificação da mesma.

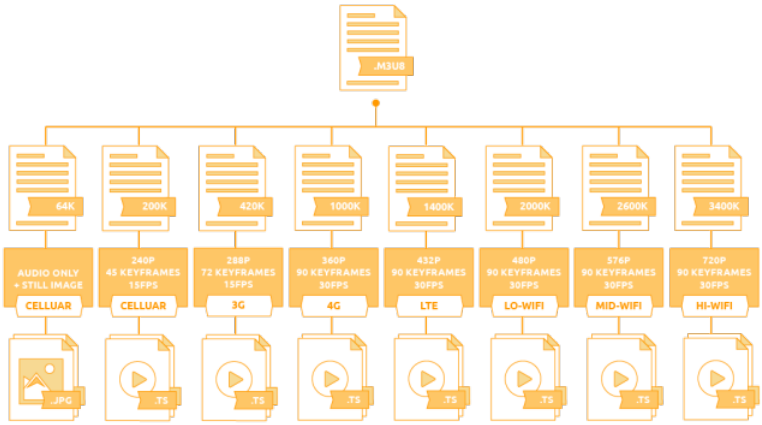

Figura 5 – Esquema de segmentação do protocolo HLS.

Fonte: https://www.encoding.com/http-live-streaming-hls/.

A segmentação, retratada na figura 5, compreende a divisão do vídeo em pequenos arquivos TS de mesma duração. Cada pedaço de mídia separado com sucesso é adicionado a um arquivo com extensão *M3U8*, também chamado de *playlist*.

As *playlists* são responsáveis por organizar os pedaços de forma sequencial para que o cliente consiga reproduzir a mídia corretamente. Cada *playlist* contém URLs que indicam *playlists* alternativas para diferentes qualidades de conexão.

Fica então de responsabilidade do reprodutor HLS receber a *playlist* do servidor, escolher a adequada banda larga disponível, agrupar os pedaços de vídeo e reproduzi-los em sequência.

[2] Disponível em: <<https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/HTTPStreamingArchitecture/HTTPStreamingArchitecture.html>>. Acesso em 30 mai. 2018.

3.2 SISTEMAS DE SUPORTE AO CONSUMIDOR

3.2.1 Zendesk

O Zendesk é uma plataforma paga que prove serviços de atendimento ao consumidor aos seus clientes. A plataforma tem o intuito de ajudar empresas a se relacionar melhor com os usuários, criando relações mais significativas[3]. A principio o objetivo é fornecer suporte e melhorar o atendimento, para depois melhorar o relacionamento e o compromisso com o cliente.

Composto por diversos módulos o software fornece assistência em diversas áreas: suporte ao cliente; capacitação de atendentes; troca de mensagens; ligações VoIP; e relatórios sobre os atendimentos.

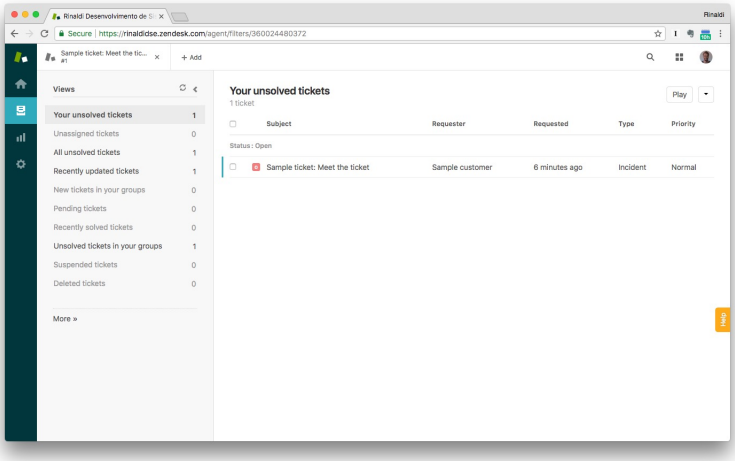

Figura 6 – Visão geral dos atendimentos

Atendimentos são chamados de *tickets* dentro da plataforma. Esses contém informações do atendimento, e estão agrupados, sejam pendentes ou resolvidos, em uma só tela de visão geral, mostrada na Figura 6.

Cada *ticket* possui suas próprias características, como prioridade, tipo e área correspondente da empresa (ver Figura 7). O contratante da plataforma pode escolher também por cadastrar um contrato SLA (*Service Level Agreement*) de resolução de problemas, que fica à mostra para que os atendentes não estourem o limite.

Se o cliente necessita de atendimento em tempo real, a empresa pode utilizar um módulo de *live chat*, por meio do qual o cliente entra em contato direto com um atendente designado pela empresa através de uma integração no site da mesma (ver Figura 8).

Apesar de esse atendimento acontecer somente por meio de troca de mensagens, outro módulo funciona com VoIP, porém não existem chamadas por vídeo.

[3] Disponível em: <https://www.zendesk.com/about>. Acesso em 20 mai. 2018.

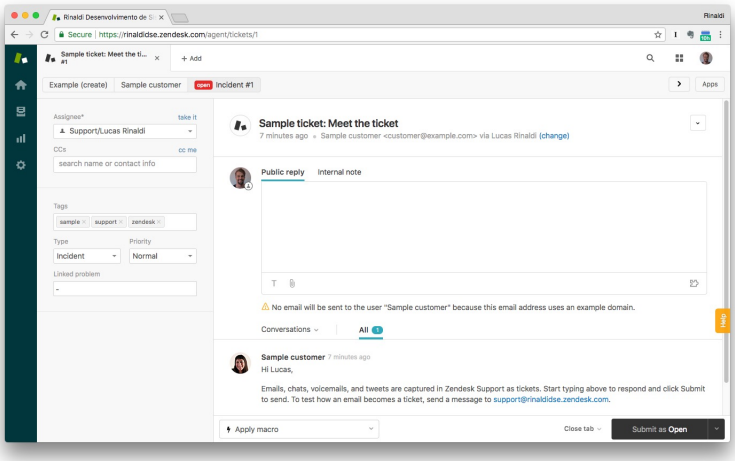

Figura 7 – Visão interna do atendimento

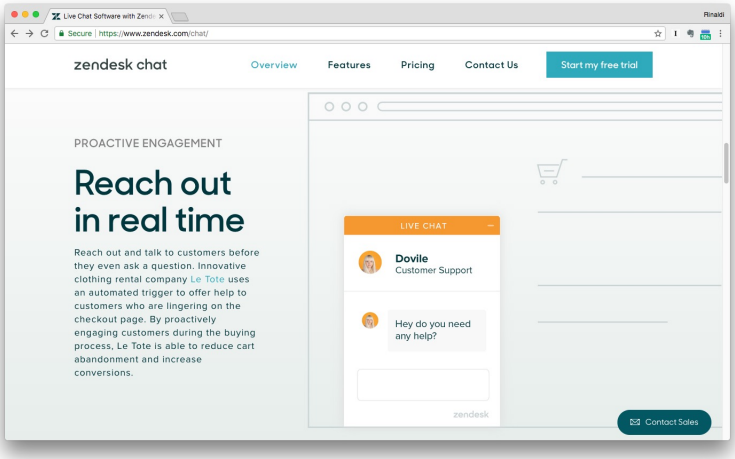

Figura 8 – Integração de chat no site do cliente

Muito utilizado, Zendesk está presente em diversas *startups* e grandes empresas, como AirBnB e OLX. Podemos ver mais exemplos na sua página de clientes[4]. A plataforma serviu de base para as primeiras ideias do projeto.

3.2.2 Intercom

Bastante voltada para soluções que colocam a troca de mensagens em foco[5]. Intercom possui múltiplos produtos que resolvem diferentes problemas nas empresas, todos com o objetivo de oferecer suporte e ajudar seus usuários diretos a reterem clientes.

O modelo de múltiplos produtos dá ao cliente oportunidade de escolher cada um separadamente. São eles *Messenger*, *Inbox* e *Articles*.

[4] Disponível em: <https://www.zendesk.com/why-zendesk/customers>. Acesso em 20 mai. 2018.
[5] Disponível em: <https://www.intercom.com/about>. Acesso em 30 jun. 2017.

Nota-se que diferentemente do Zendesk, a empresa direciona grande parte de seus esforços a área de retenção de clientes. Coloca-os em etapas de funil de vendas e utilizando o seu produto *Messenger* envia mensagens de acordo com a etapa atual. No projeto apresentado nesse trabalho não temos um foco em marketing ou vendas, por isso a pequena explicação.

Como citado acima, seus produtos prezam pela troca de mensagens. O módulo *Inbox* funciona como um live chat integrado no site do usuário, conectando assim o seu cliente com atendentes de suporte.

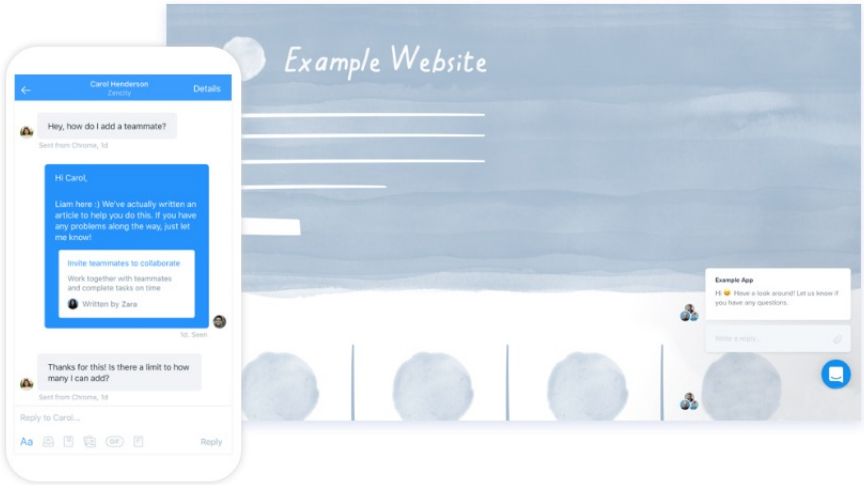

Figura 9 – Intercom Inbox

Sua vantagem é possuir um conexão facilitada com *Articles*, seu outro produto, e assim fazer com que o atendente resolva os problemas de forma rápida e fácil.

Articles é uma central de ajuda, onde guias sobre o *software* e perguntas mais frequentes podem ser publicadas para melhor resolução de problemas.

A integração com o *Inbox* serve para que o seu time consiga escalar o suporte. No momento que o atendente recebe uma palavra-chave a central de ajuda retorna uma lista de artigos relacionados à mesma.

Traz também relatórios e ações sobre como os clientes estão utilizando a central e como melhora-lá. Na Figura 11 vemos exemplos de ações a serem tomadas.

Centrado e com produtos voltados para o marketing algumas funcionalidades divergem do objetivo desse projeto, porém o Intercom serviu de inspiração para ideias futuras.

A integração da central de ajuda dentro do chat é um recurso poderoso que poderia ser integrado dentro de uma videoconferência. Em tempo real o atendente pode pesquisar sobre o assunto que o cliente está falando e indicar a resposta certa, facilitando a resolução do problema.

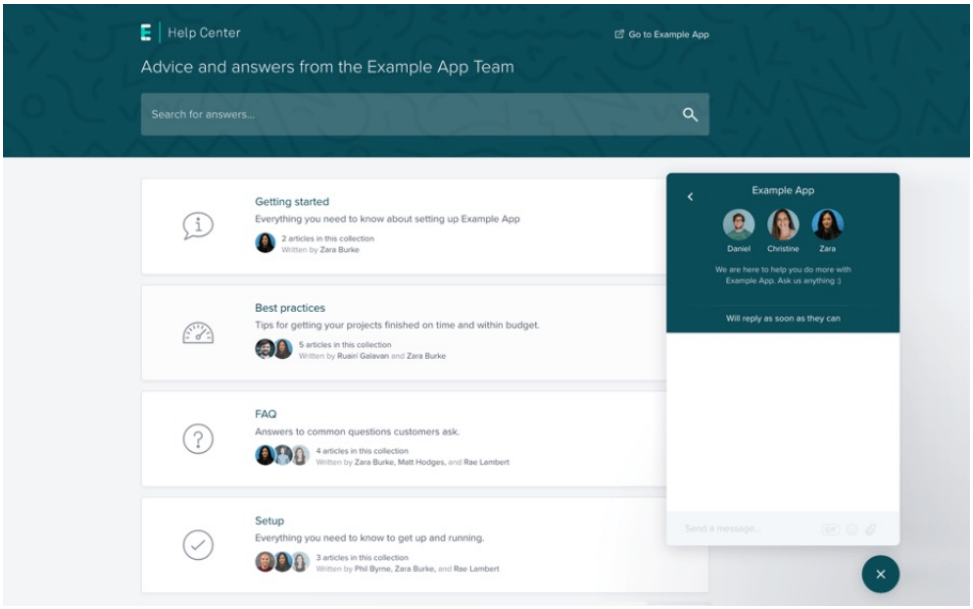

Figura 10 – Intercom Articles

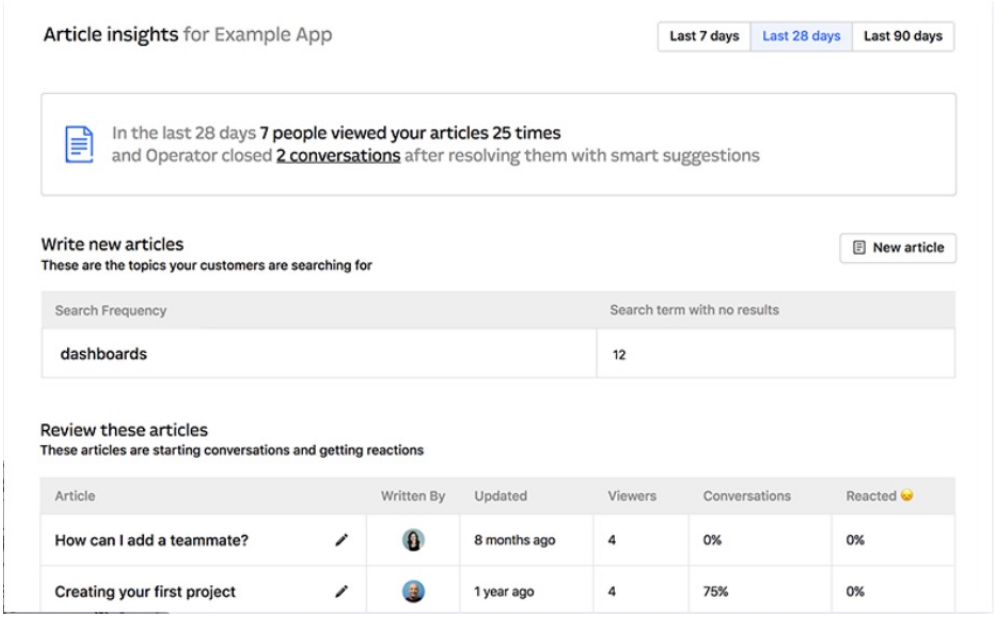

Figura 11 – Relatório e ações no Intercom Articles

4 PROJETO

Nesse capítulo apresentamos a estrutura, arquitetura e funcionalidades do projeto abordado realizado no trabalho. Em primeiro momento foi feita uma análise dos requisitos necessários, para que então pudéssemos organizar todas as funcionalidades dentro do protótipo da aplicação e por fim projetar a arquitetura do *software* e a modelagem do banco de dados.

4.1 ANÁLISE DE REQUISITOS

Dois tipos de requisitos foram levantados para a implementação do projeto: Requisitos Funcionais (RF), que são aqueles que definem as funcionalidades e como o *software* se comporta; e os Requisitos Não-Funcionais (RNF), que são responsáveis por especificar quesitos mais técnicos do *software*, como tecnologias, segurança e desempenho.

Para definição dos RF, foram analisados programas no estado da arte das duas categorias abordadas (videoconferência e atendimento ao consumidor) e coletadas funcionalidades consideradas importantes. Os RNF foram definidos a partir de estudos de aplicações Web modernas, com exceção do WebRTC que foi definido desde o principio. Requisitos funcionais serão apresentados com um diagrama de casos de uso, sendo cada caso um requisito.

Os RNF levantados são os seguintes (os itens 11, 12 e 13 foram baseados no trabalho [Fosser e Nedberg 2018, Sec. 4.3.1]):

- RNF 01: Utilizar a linguagem JavaScript.
- RNF 02: Arquitetura REST para API.
- RNF 03: Arquitetura com WebSockets para estabelecer a conexão P2P.
- RNF 04: Utilizar plataforma NodeJS para o servidor.
- RNF 05: Utilizar *framework* ExpressJS para API.
- RNF 06: Utilizar *framework* SocketIO para suporte a WebSocket.
- RNF 07: Utilizar *framework* ReactJS para renderizar interface no navegador.
- RNF 08: Utilizar PostgreSQL como gerenciador de banco de dados.
- RNF 09: Suporte a dispositivos móveis.

- RNF 10: Utilização de WebRTC para videoconferência.
- RNF 11: A taxa de transmissão deve estar acima de 1.0 Mbps.
- RNF 12: Porcentagem de perda de pacotes não deve ultrapassar o máximo de 10%.
- RNF 13: Limite da latência da comunicação de 1000 milisegundos.

Os RF levantados são representados como casos de uso na Figura 12.

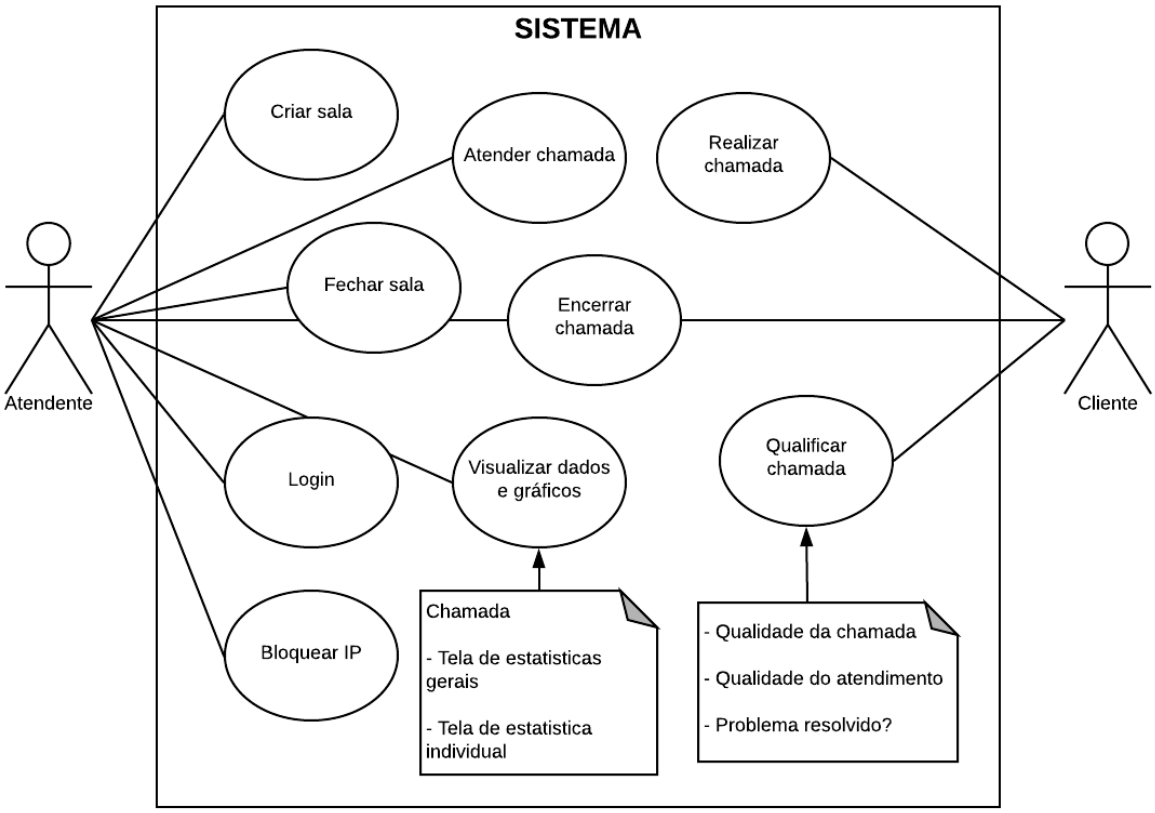

Figura 12 – Requisitos Funcionais do sistema representados por casos de uso

4.2 PROTÓTIPOS DE TELA

4.2.1 Login

A tela inicial da aplicação é a de login (ver Figura 13), constituída de um pequeno formulário de autenticação e um logo no canto esquerdo.

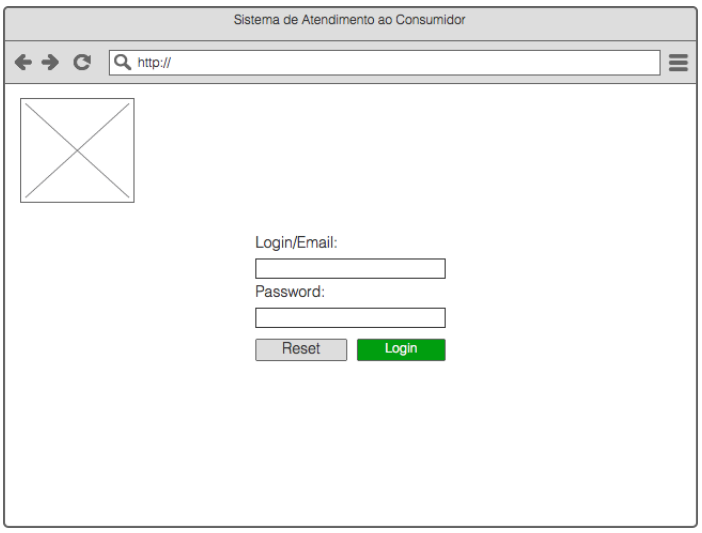

Figura 13 – Tela de login

Como nenhum cliente necessita de cadastro para realizar chamadas, a tela está disponível somente para os atendentes entrarem no gerenciamento de tickets.

Ao inserir os dados corretamente o atendente é redirecionado para a lista de tickets (ver 4.2.2). Caso contrário, um erro é mostrado no formulário.

4.2.2 Gerenciamento de tickets

A tela de Gerenciamento de Tickets é a tela acessada logo após a autenticação do atendente no sistema de atendimento. Podemos observar nas figuras 14, 15 e 16 que a interface é constituída de uma barra lateral na esquerda e o conteúdo principal à direita. O conteúdo a ser mostrado na direita dependerá do contexto da aplicação e de onde o usuário está no momento.

De inicio, a tela de estatísticas de tickets relacionados ao atendente (Figura 14). Essa tela aparece no conteúdo principal quando o usuário não tem nenhum ticket selecionado. Ela reúne informações de todos os atendimentos feitos pelo usuário autenticado, como:

- Número total de chamadas;
- Tempo total de chamada;
- Média de qualidade de atendimento;

- Média de qualidade de ligação; e
- Porcentagem de problemas resolvidos.

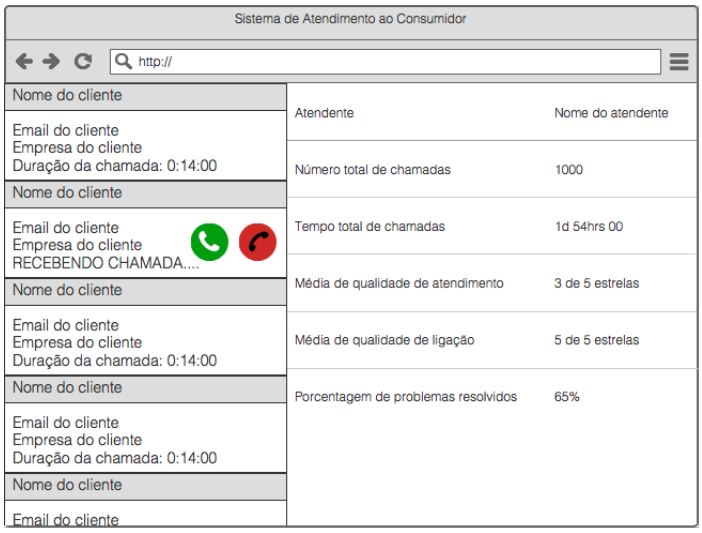

Figura 14 – Tela de estatísticas do atendente

A segunda tela (Figura 15) diz respeito a informações sobre um ticket específico que está selecionado.

É importante frisar que o item selecionado é de um atendimento já encerrado, pois o conteúdo é diferente e mostra informações adicionais sobre a qualidade da chamada e algumas vezes até a opinião do cliente atendido, se fornecida pelo mesmo.

Os dados presentes são:

- Nome do cliente e e-mail do cliente;
- Empresa, se existir;
- Descrição do problema;
- Duração do atendimento;
- Qualidade da ligação;
- Qualidade do atendimento; e se
- O problema foi resolvido.

Por último, temos na Figura 16 a tela que mostra o conteúdo de um ticket em aberto, ou seja, que o cliente está requisitando a chamada no momento. Nessa tela são exibidas as seguintes informações:

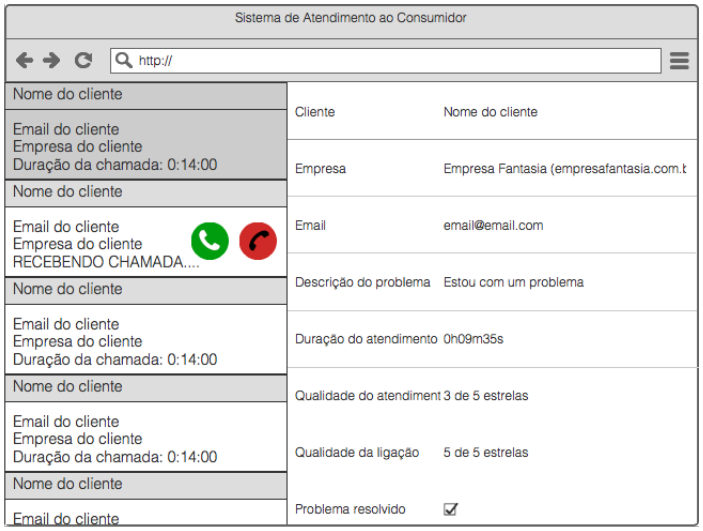

Figura 15 – Tela de estatísticas de um ticket encerrado

- Nome e-mail do cliente;
- Empresa, se existir;
- Descrição do problema; e
- Botões de aceitar ou recusar chamada.

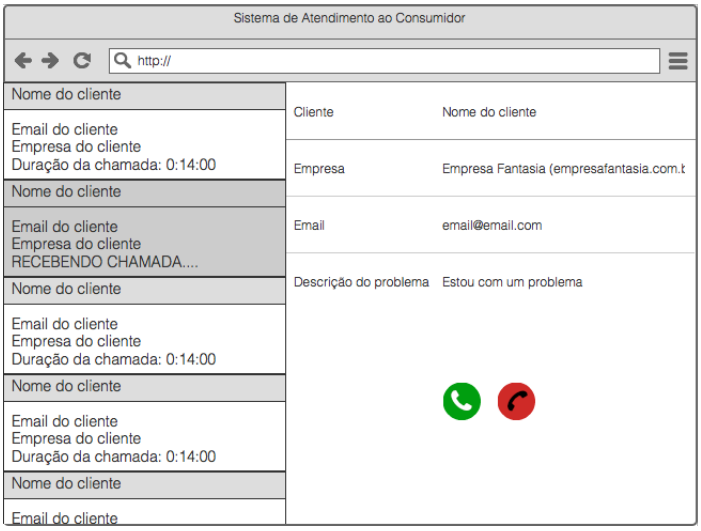

Figura 16 – Tela de estatísticas de uma requisição de chamado

4.2.3 Plugin para videoconferência

O sistema de atendimento fornece aos seus usuários uma maneira de integrar um videochat no site da empresa através de um plugin. Esse plugin é feito a partir de

tecnologias disponíveis no navegador (ou seja, não requer outros plugins, como Flash). O chat fornece três interfaces diferentes:

- Plugin fechado e com um botão para abrir localizado no canto direito inferior;
- Plugin aberto com um formulário de requisição de atendimento; e
- Plugin aberto após encerramento de chamada com botões para qualificar atendimento.

4.2.4 Tela de videoconferência

Consiste na interface que compreende a funcionalidade principal do sistema de atendimento. Foi separada em um módulo e está presente tanto na aplicação voltada para atendentes quando no videochat integrado no site do cliente. A tela contém os seguintes itens:

- Imagem da câmera do cliente sendo atendido;
- Imagem da câmera do atendente prestando atendimento;
- Botões para encerrar chamada e silenciar microfone;
- Dados sobre o presente atendimento.

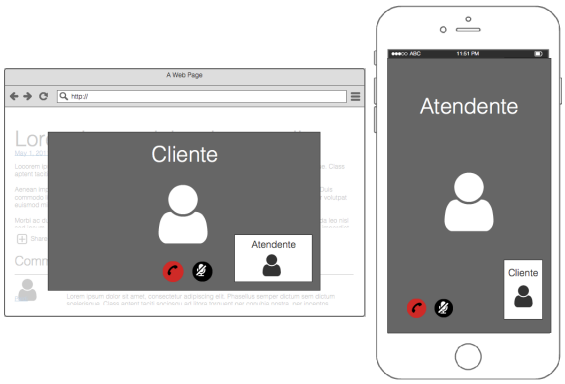

Figura 17 – Videoconferência entre atendente e cliente

4.3 ARQUITETURA

Observamos na Figura 18 uma visão geral da arquitetura do sistema. Notamos que o modelo utilizado é o clássico cliente-servidor.

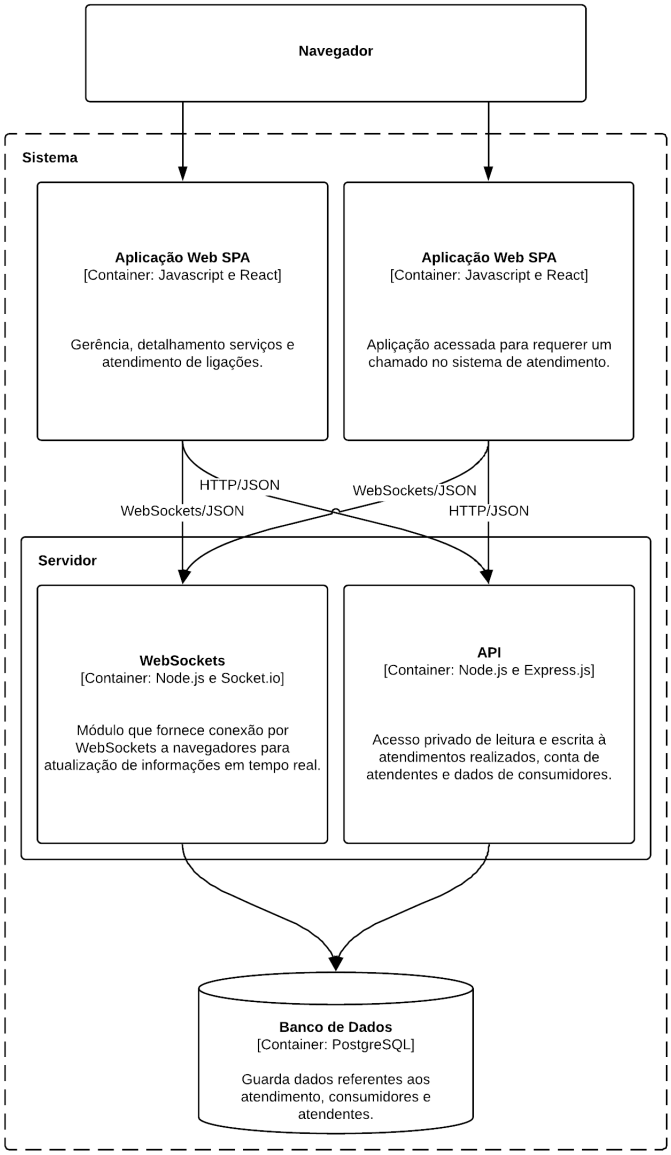

Figura 18 – Arquitetura do sistema

4.3.1 Servidor

Todo o *back-end* foi implementado sobre a plataforma NodeJS. Essa tornou-se popular dentre os desenvolvedores nos últimos tempos devido à capacidade de executar código JavaScript fora do navegador.

Podemos elencar três componentes mais importantes no lado do servidor. O primeiro deles é a base, uma interface de programação de aplicação (API) nos modelos da arquitetura REST, implementada com o *framework* ExpressJS, utilizado para facilitar a construção de aplicações Web. O servidor da API, além de ser utilizado para salvar informações no banco de dados, também é responsável por iniciar a conexão através de *sockets*.

O segundo componente é outro *framework*, SocketIO, desenvolvido com o o objetivo

de facilitar o *upgrade* de requisições HTTP e estabelecer a comunicação através de WebSockets. Esse *framework* abstrai as APIs complexas do navegador em padrões já conhecidos, fornecendo ao desenvolvedor uma outra arquitetura mais simples de usar.

Por último, temos o gerenciador de banco de dados PostgreSQL, que armazena os dados das chamadas, atendentes, administradores e clientes do sistema.

4.3.2 Cliente

O lado do cliente será sempre executado em um navegador, ou seja, foi inteiramente implementado com HTML e JavaScript.

Utiliza o *framework* React para renderizar interfaces. É baseado em composição de componentes, sendo cada um deles uma peça da interface. Esses componentes têm acesso às APIs nativas do navegador, podendo assim realizar requisições HTTP através de interfaces como AJAX.

É responsabilidade do cliente solicitar uma conexão peer-to-peer com um navegador remoto, conforme ilustrado na figura 19, que remete a assuntos abordados no capítulo 2.

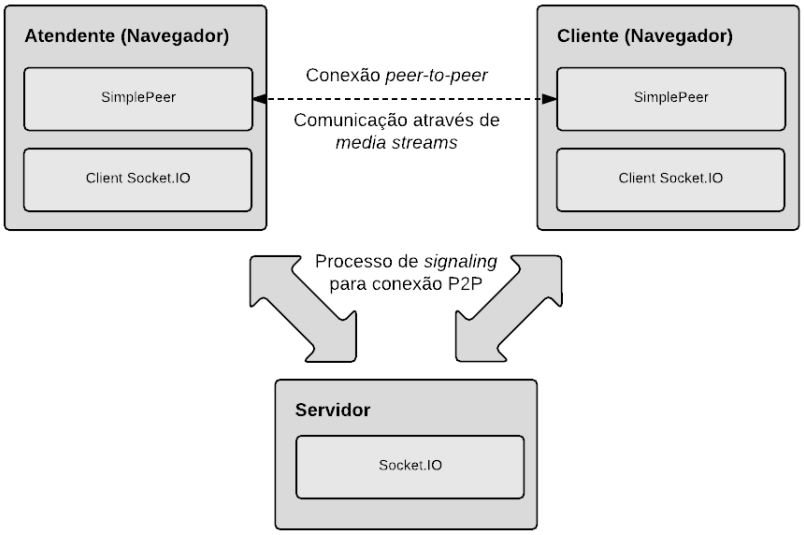

Figura 19 – Conexão peer-to-peer através de WebSockets

Com a conexão estabelecida entre os dois pontos, a aplicação renderiza automaticamente uma nova tela, na qual o cliente consegue enxergar o atendente e vice-versa.

4.4 MODELAGEM DO BANCO DE DADOS

O diagrama de entidades, representado na figura 20, contém a modelagem do banco de dados da aplicação.

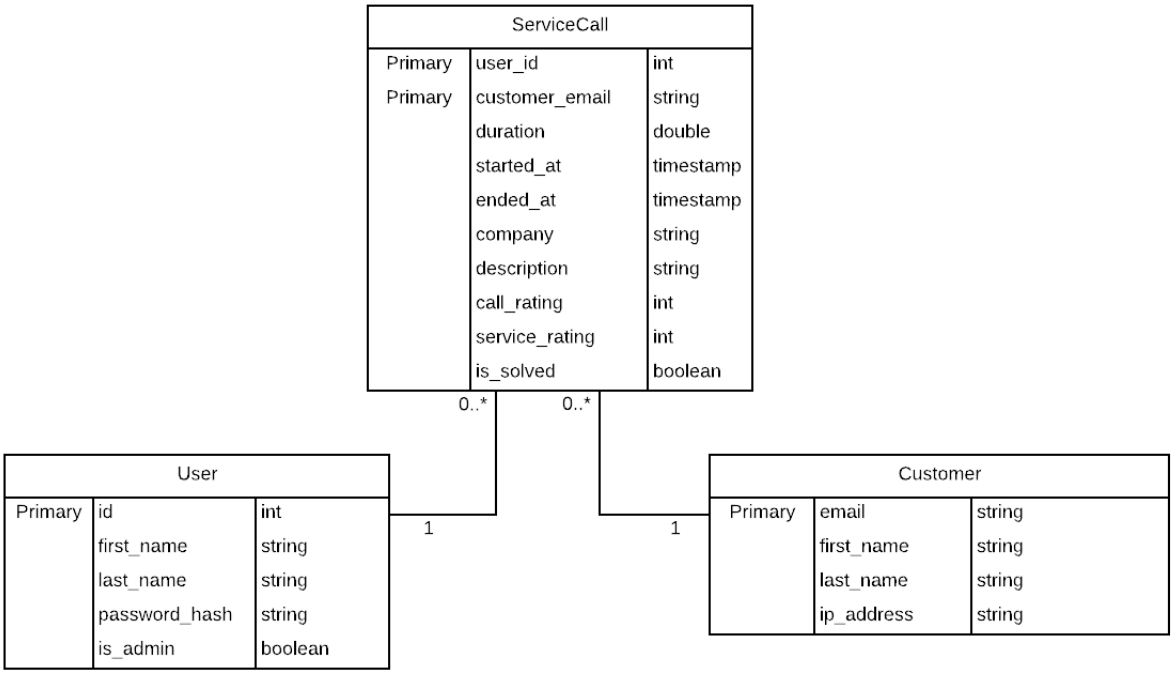

Figura 20 – Entidades do banco de dados

A entidade *User* representa os atendentes do serviço de atendimento ao consumidor. Todo atendente necessita fazer login no sistema para poder criar salas e atender clientes. Esse mesmo usuário tem permissão para visualizar dados e estatísticas de chamadas.

Para representar os clientes que realizam chamadas de suporte criamos a entidade *Customer*. Essa entidade não necessita de login para solicitar um atendente, visto que o objetivo do sistema é prover uma maneira fácil para pessoas obterem resolução de problemas.

Por fim, as chamadas de suporte são representadas pela entidade *ServiceCall*. Essa entidade possuí o tempo de duração de atendimento e faz a relação entre atendente e consumidor. Foram adicionados também dados da chamada, como avaliação e se o problema foi resolvido ou não, para que seja possível registrar estatísticas e criar uma página com informações para os atendentes.

5 IMPLEMENTAÇÃO

5.1 TECNOLOGIAS E FERRAMENTAS

5.1.1 JavaScript

No início da Internet, os únicos documentos transferidos eram páginas estáticas HTML. Nenhuma página possuía conteúdo dinâmico e nem possibilidade de interação com a mesma. Javascript foi idealizado com esse intuito. Desenvolver uma forma com que os usuários possam realizar ações e receber resultado de forma mais rápida. A web precisava ser mais dinâmica, de acordo com Marc Andressen, fundador da Netscape Communications[1].

Criada em 1995, no meio da chamada *browsers war* (ou guerra dos navegadores). Foi desenvolvida em apenas dez dias por Brendan Eich, e a princípio levou o nome *Mocha*. Em agosto do mesmo ano, a Netscape em parceria com a Sun Microsystems (criadora da linguagem Java, posteriormente adquirida pela Oracle) apresentou o JavaScript[2], com o intuito de torná-la a linguagem de script oficial dos navegadores.

Javascript na verdade é uma implementação/extensão da especificação ECMAScript ([International 2017]), desenhada para facilitar a padronização de linguagens proprietárias baseadas na ECMAScript, como por exemplo JScript, a linguagem de script da Microsoft. Essa especificação define uma linguagem com paradigma orientado a objetos, tipagem fraca e que funciona através de interpretadores, ou seja, não compilada (diferentemente de Java, apesar do nome).

A grande diferença entre a especificação e a implementação são as interfaces de aplicação disponíveis na linguagem executada pelos navegadores. Como por exemplo, a API voltada para interação com HTML, chamada de DOM ou *Document Object Model*, que foi a primeira responsável pelo maior dinamismo das páginas da Internet. Hoje em dia temos interfaces para captura de mídia (áudio, vídeo); interação com dispositivos móveis (bateria, GPS); e até para conexões *peer-to-peer*, assunto do presente trabalho.

Cada vez mais utilizada devido a sua presença na Web, muitos desenvolvedores acreditaram que a linguagem era digna de funcionar também no servidor, eliminando a necessidade de navegadores. Com o lançamento da *engine* V8, o interpretador desenvolvido pela Google para ser utilizado no seu navegador Chrome, a linguagem atingiu um outro nível de utilização, passando a ser utilizada também em servidores, através do ambiente

[1] Disponível em: <https://auth0.com/blog/a-brief-history-of-javascript/>. Acesso em 20 mai. 2018.
[2] Disponível em: <http://tech-insider.org/java/research/1995/1204.html>. Acesso em 20 mai. 2018.

Node.js. Denominado *cross platform*, o Node.js é capaz de ser executado em qualquer sistema operacional e funciona em cima do interpretador V8, tornando a linguagem livre das amarras dos navegadores. A linguagem também pode ser encontrada em aplicações de banco de dados não-relacionais, como o MongoDB.

Nesse projeto, JavaScript será utilizado tanto no cliente como no servidor (com exceção do banco de dados). Devido a isso, o primeiro item do tópico de tecnologias foi uma introdução à mesma. Abordaremos nos próximos itens como esta linguagem será utilizada no projeto.

5.1.2 Servidor

O Servidor é constituído por uma API e uma conexão com WebSockets. A interface REST é usada para enviar e receber dados de sobre os atendimentos do banco de dados. O uso de WebSockets permite realizar as conexões P2P nas videoconferências.

São utilizados *frameworks* e bibliotecas desenvolvidos com a plataforma NodeJS. O gerenciador de pacotes NPM (*Node Package Manager*) foi usado para instalação das dependências do projeto.

ExpressJS é a ferramenta utilizada para criação da API REST que servirá para enviar e receber dados do banco de dados sobre os atendimentos. Responsável também por devolver uma página HTML a ser integrada no cliente.

SocketIO fornece uma interface de aplicação em cima de uma conexão por WebSockets e auxiliará as conexões *peer-to-peer*.

O servidor possui também uma biblioteca, chamada ObjectionJS, que efetua o mapeamento objeto-relacional, tornando mais fáceis as operações no banco de dados.

5.1.2.1 ExpressJS

ExpressJS é um dos primeiros e mais famosos *frameworks* desenvolvidos para servir aplicações Web com NodeJS. Tornou-se popular pela sua facilidade de uso, alta possibilidade de customização e suporte a ferramentas de linha de comando. Observamos em seu site[3] o foco em se manter como uma ferramenta minimalista e *unopinionated*, que tem como objetivo dar mais controle aos desenvolvedores, diferentemente de grandes *frameworks* como Ruby On Rails.

Possui uma arquitetura voltada a *middlewares*[4]. Esse tipo de arquitetura suporta o registro de módulos que são executados em ordem a cada ciclo de requisição e resposta HTTP. Devido a sua natureza minimalista, a maior parte das customizações do servidor

[3] Disponível em: <http://expressjs.com/>. Acesso em 15 mai. 2018.

[4] Disponível em: <https://www.cleveroad.com/blog/the-best-node-js-framework-for-your-project–express-js–koa-js-or-sails-js>. Acesso em 20 mai. 2018.

são feitas através dos *middlewares*. Esses são responsáveis por adições de cabeçalhos HTTP, abertura de conexão com o banco de dados, autenticação de usuário, dentre outras.

Todas as funcionalidades presentes no ExpressJS poderiam ser implementadas com NodeJS puro. De acordo com 5.1, o *framework* traz aos usuários ferramentas para desenvolver servidores HTTP com rapidez e simplicidade, permitindo implementar um servidor com apenas algumas linhas de código:

```
1 var express = require('express');
2 var app = express();
3 app.get('/', function(req, res) {w
4   res.send('<h1>Hello World!</h1>');
5 });
6 app.listen(8000);
```

Listing 5.1 – Servidor ExpressJS

Toda essa simplicidade e liberdade de customização vem com certos custos. A aplicação de um time pode acabar muito diferente de outros, o que não acontece em *frameworks* com padrões bastante definidos, como por exemplo Django, escrito em Python.

Devido a sua agilidade de desenvolvimento, utilizaremos ExpressJS para construir uma API RESTful, responsável por salvar e trazer os dados dos usuários conectando com o banco de dados através de uma biblioteca.

5.1.2.2 SocketIO

SocketIO é um *framework* escrito em JavaScript voltado para aplicações web em tempo real.

A arquitetura é baseada em um sistema com padrão baseado em eventos que funciona de forma bi-direcional, ou seja, do navegador ao servidor e vice-versa. É constituída de um servidor implementado com Node.js e uma biblioteca JavaScript que realiza a conexão a partir do cliente.

Para efetuar a comunicação em tempo real, os contribuidores do projeto desenvolveram o protocolo SocketIO[5] que foi em seguida implementado pelo *framework*. O documento descreve uma arquitetura que realiza o *Upgrade* de uma conexão HTTP para WebSockets, caso suportado. Caso não haja suporte a WebSockets, usa como *fallback* outras alternativas, como *long-polling*.

[5] Disponível em: <https://github.com/socketio/socket.io-protocol>. Acesso em 24 mai. 2018.

Apesar de utilizar conexões WebSockets (quando possível), o *framework* não é uma implementação da especificação. Ele age como uma interface de aplicação, encapsulando uma *engine*/biblioteca que funciona de acordo com o protocolo WebSocket, como podemos ver no relatório de mudanças da versão 2.0[6].

Ao invés de usarmos somente uma biblioteca que implemente o protocolo WebSockets ou logo a interface nativa do navegador, optamos por essa ferramenta mais robusta devido a algumas funcionalidades descritas na página de documentação[7]:

- Suporte a *multiplexing*: facilita a comunicação em diferentes canais, porém utilizando a mesma conexão;
- Confiabilidade: consegue conectar-se mesmo através de *proxies* e *firewalls*; e
- Suporte a reconexão automática em caso de perda de conexão.

Observamos na listagem 5.2 como é simples a integração com um servidor NodeJS. Iniciamos o servidor executando a função *createServer* onde registramos uma aplicação ExpressJS. Na quarta linha é criado um objeto *io* que atua como um barramento de eventos, escutando por eventos *connection*, que no projeto utilizaremos para realizar as conexões P2P.

```
1 var express = require('express');
2 var app = express();
3 var server = require('http').createServer(app);
4 var io = require('socket.io')(server);
5 io.on('connection', function(){ /* */ });
6 server.listen(3000);
```

Listing 5.2 – Servidor ExpressJS integrado com SocketIO

O projeto integrará o *framework* no servidor para troca de informações sobre *peers* de uma videoconferência, para que então os navegadores possam realizar a conexão P2P entre eles e eliminar a necessidade do servidor.

5.1.2.3 ObjectionJS

ORM ou *Object-Relational Mapping* é uma técnica relacionada a manipulação e conversão de dados. Consiste em mapear elementos de um banco de dados, como tabelas, colunas e linhas para o paradigma orientado a objetos.

[6] Disponível em: <https://socket.io/blog/socket-io-2-0-0/>. Acesso em 24 mai. 2018.
[7] Disponível em: <https://github.com/socketio/socket.io/#features>. Acesso em 24 mai. 2018.

Geralmente quando empregamos o termo ORM, estamos nos referindo uma biblioteca que implementa essa técnica, encapsulando o código responsável por fazer consultas, inserções e atualizações no banco, e abstraindo o programador de linguagens de acesso a bancos de dados, como SQL.

Para esse projeto foi escolhida uma biblioteca chamada ObjectionJS[8], desenvolvida para servidores NodeJS e que integra-se facilmente com o *framework* ExpressJS. Com essa ferramenta conseguimos agilizar o processo de desenvolvimento através de mapeamentos das entidades do banco de dados (como no exemplo da Listagem 5.3) para modelos no servidor, evitando a utilização de códigos SQL e aproveitando técnicas com *lazy loading* fornecidas pela biblioteca.

```
1  const { Model } = require('objection');
2
3  export class User extends Model {
4    static tableName = 'users';
5    static idColumn = 'id';
6
7    fullName() {
8      return this.name + ' ' + this.lastName;
9      }
10 }
```

Listing 5.3 – Mapeamento modelo User

ObjectionJS tem suporte testado e garantido para três gerenciadores de banco de dados, que são SQLite3, PostgreSQL e MySQL. Devido à sua grande popularidade, utilizaremos o segundo da lista, PostgreSQL.

5.1.3 Cliente

Os navegadores atuais suportam as seguintes tecnologias para renderização de páginas: HTML, CSS e JavaScript. Com a popularidade da Web, surgiram comunidades e empresas interessadas pelo avanço da Internet. Em conjunto, surgiram órgãos responsáveis pela definição de especificações e padrões de cada uma dessas tecnologias, como por exemplo a W3C (*World Wide Web Consortium*).

Devido à grande diversidade de *browsers* existentes e cada um deles apostando em sua própria implementação, o avanço dos navegadores ficou estagnado e anda relativamente devagar em comparação a outras ferramentas de desenvolvimento.

[8] Disponível em: <https://vincit.github.io/objection.js/#introduction>. Acesso em 29 mai. 2018.

Com essa demora para atualização dos *standards*, desenvolvedores decidiram agilizar o processo e utilizar recursos que ainda não foram padronizados. Existem diversas bibliotecas, ferramentas de linha de comando e empacotadores de módulos que *transpilam*[9] uma versão mais nova do JavaScript para o suportado oficialmente pelos navegadores.

A parte que compreende o cliente na nossa arquitetura envolve tecnologias que auxiliam esse avanço.

5.1.3.1 Webpack

Navegadores atualmente não suportam oficialmente o conceito de módulos JavaScript. Existem duas opções para melhor organizar o código em grandes projetos: dividir o código em vários arquivos de script e utilizar mais de uma chamada no servidor para executá-los na página; ou agrupar todo o código em um só arquivo e realizar somente uma requisição HTTP.

A segunda opção é a mais difundida em projetos com uma estrutura complexa. No navegador, para que a separação de módulos seja bem sucedida, a ordem de importação dos scripts deve ser respeitada. Adicionalmente, dependendo da quantidade, a importação pode impactar bastante na performance da aplicação, devido à abertura de diversas conexões HTTP. Reunir todos os módulos necessários e importá-los na ordem correta manualmente não é viável para times ágeis que prezam pela produtividade e a escalabilidade de um produto.

Para atingir esse objetivo, foi utilizado o empacotador de módulos Webpack[10]. Em sua configuração mais simples, o Webpack necessita de um arquivo de configuração na raiz do projeto que indique um ponto (arquivo) de entrada e a pasta de saída, onde será armazenado o resultado do empacotamento. Esse resultado, normalmente referido como *build*, é constituído de arquivos CSS, JavaScript e uma página HTML responsável por importar e executar os scripts.

5.1.3.2 ReactJS

JavaScript é por design uma linguagem orientada a objetos e imperativa. Com o aumento da popularidade de linguagens funcionais na comunidade, surgiram diversas bibliotecas que trazem técnicas funcionais para o JavaScript, incluindo ReactJS.

O ReactJS é uma biblioteca JavaScript desenvolvida pelo Facebook voltada para construção de interfaces, ou seja, compreende o V em um arquitetura MVC (*Model-View-Controller*).

[9] Compilar de uma linguagem para outra, porém no mesmo nível de abstração
[10] Disponível em: <https://webpack.js.org/>. Acesso em 29 mai. 2018.

ReactJS introduziu uma forma de escrever interfaces quando apresentou uma nova linguagem de script que transpila para JavaScript puro, chamada JSX. JSX possibilita o uso de uma sintaxe similar à linguagem de marcação HTML (declarativa por design) dentro de scripts JS.

```
1  function formatName(firstName, lastName) {
2    return firstName + ' ' + lastName;
3  }
4
5  function Component({ firstName, lastName }) = {
6    return (
7      <h1>
8      Hello, {formatName(firstName, lastName)}!
9      </h1>
10   );
11 }
12
13 render(
14   <Component firstName="John" lastName="Doe" />
15 );
```

Listing 5.4 – Exemplo de componente escrito com JSX

No código 5.4 observamos o uso tanto de *tags* HTML, quanto de funções imperativas. A função *formatName* é padrão da linguagem JavaScript, enquanto a *Component* define um dos conceitos mais importantes da biblioteca ReactJS, o componente.

Outra característica semelhante a linguagens funcionais é a ideia de funções puras[11]. Uma função normal pode ser pura se: para o mesmo argumento de entrada sempre retornar o mesmo resultado; e não produzir efeitos colaterais, ou seja, alterar algum dado fora do seu escopo interno. Esse tipo de função é popular devido a sua previsibilidade, clareza e facilidade de testar.

A biblioteca tem uma arquitetura baseada no desenvolvimento orientado a componentes, o que significa que as interfaces resultantes de projetos ReactJS são construídas a partir de composições dos mesmos. Esses componentes são as funções puras da biblioteca e idealmente devem ser encapsulados, com objetivo definido e retornar o mesmo resultado sempre que os atributos[12] declarados sejam iguais.

ReactJS foi escolhido para o desenvolvimento da interface devido a sua popularidade

[11] Disponível em: <https://medium.com/javascript-scene/master-the-javascript-interview-what-is-a-pure-function-d1c076bec976>. Acesso em 29 mai. 2018.

[12] *firstName* e *lastName* em 5.4

na comunidade, pela sua facilidade de uso, apesar de um alta curva de aprendizado e o aspecto funcional e declarativo.

5.2 DESENVOLVIMENTO DO SISTEMA

Com base na modelagem de classes e nos casos de uso levantados foi criada uma arquitetura para suprir as necessidades do sistema e do problema apresentado no trabalho. O projeto foi desenvolvido utilizando Git para controle de versão junto do sistema Github para armazenar o código em um repositório online.

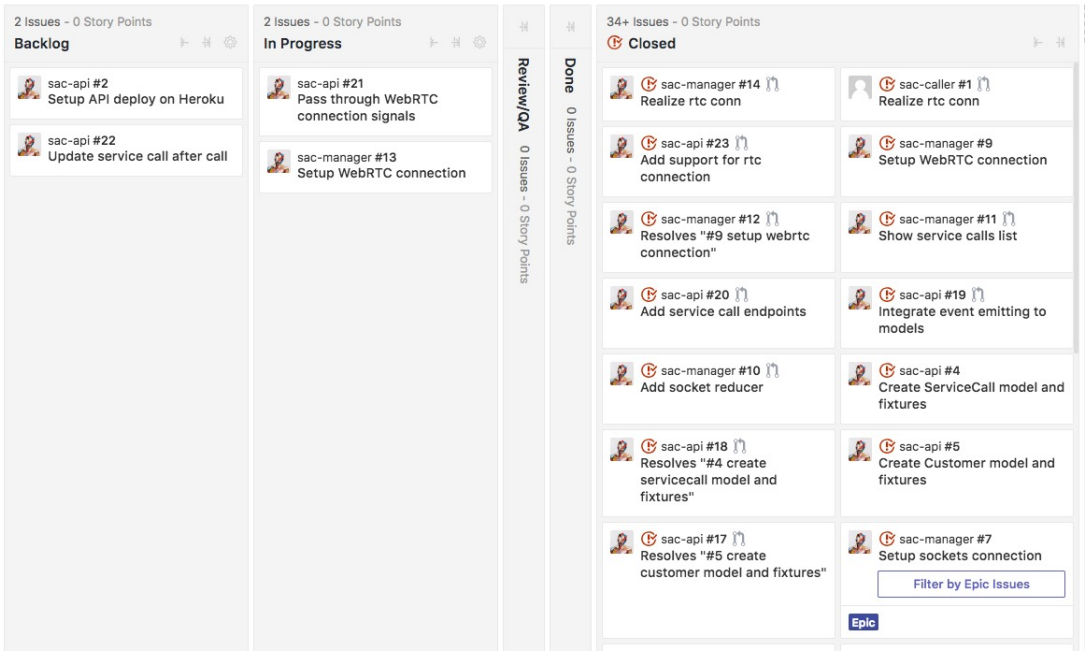

Figura 21 – Organização do projeto dentro do Github/ZenHub

Foi utilizada a extensão ZenHub (Figura 21) para organização e gerenciamento de tickets com uma metodologia similar ao Kanban. O sistema é constituido de quatro projetos, *SAC API*, *SAC Manager*, *SAC Caller* e *SAC Form*. Sua arquitetura será descrita na sequência.

5.2.1 Visão geral do sistema

O sistema de atendimento ao consumidor foi pensado de maneira a fornecer ao usuário uma facilidade ao se conectar com seu próprio cliente. A decisão de dividir o projeto em múltiplas partes veio da necessidade do *sac-caller* ser usado em dispositivos móveis, ou seja, era necessário uma aplicação pequena, que não consumisse muitos dados do pacote de internet do usuário para baixá-la.

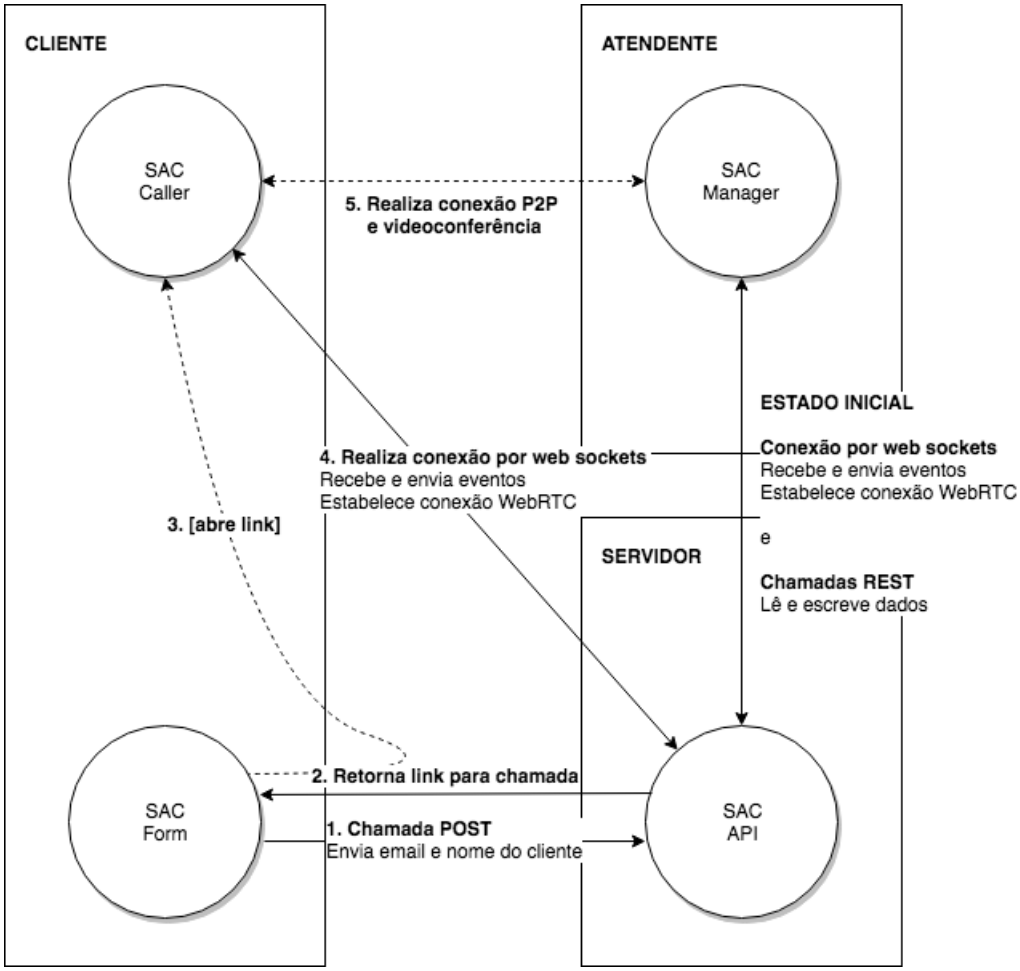

Figura 22 – Visão geral dos projetos do sistema

Observamos na Figura 22 como os componentes do sistema estão organizados. O projeto *SAC API* é um servidor com uma API REST que possui suporte a conexão *WebSockets* para a facilidade de troca de mensagens. Já SAC *Manager*, *Caller* e *Form* são os três constituídos somente de HTML, JavaScript e CSS, sendo os dois primeiros aplicações feitas em ReactJS.

Na figura Figura 23 está explicado como é o fluxo para realizar uma conexão entre cliente e usuário:

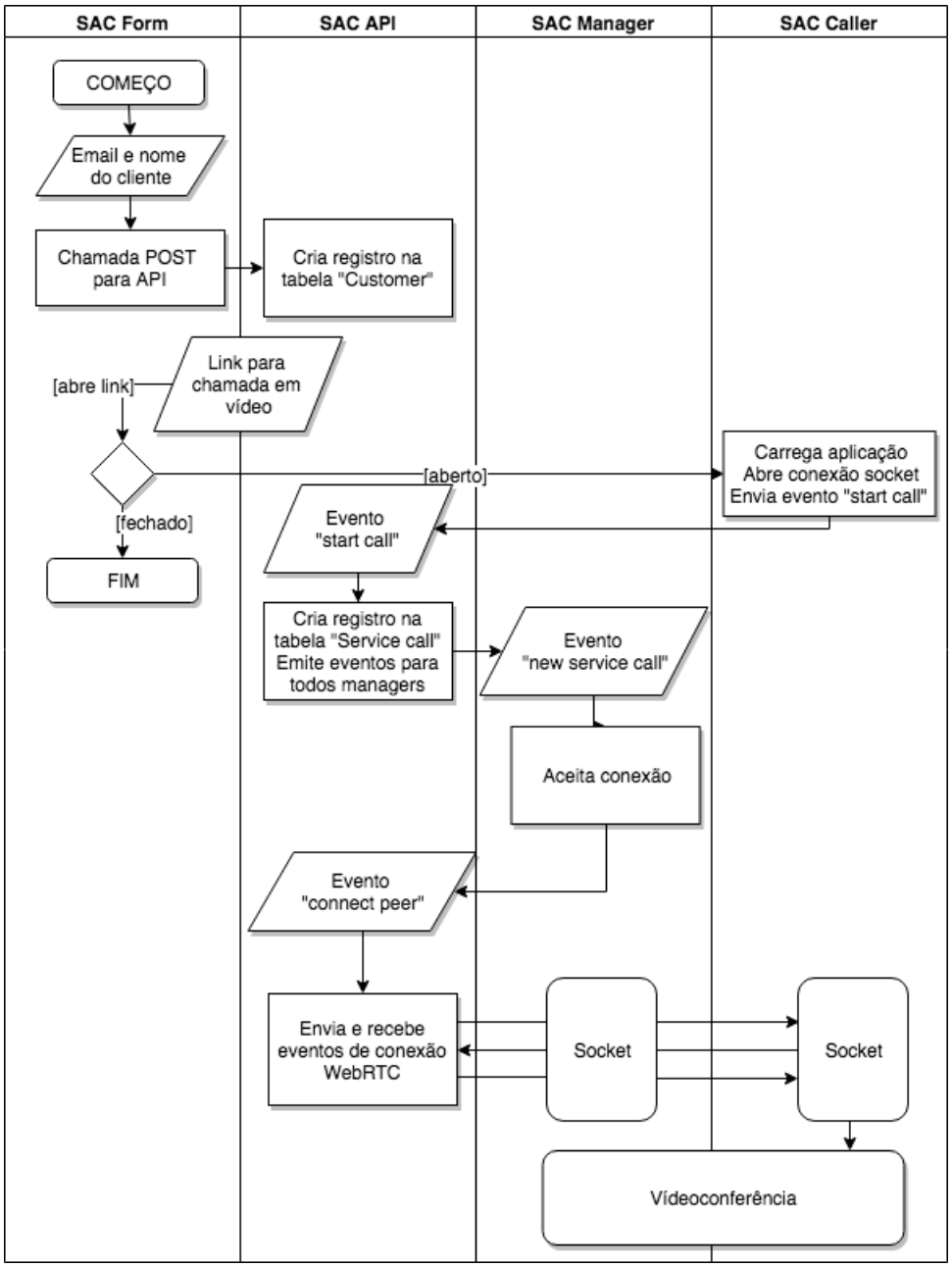

Figura 23 – Fluxograma conexão WebRTC entre cliente e usuário

5.2.2 SAC *Manager*

Esta aplicação tem o propósito de ajudar os atendentes do sistema de atendimento a receber chamadas de clientes, controlar seus tickets, checar se os problemas foram resolvidos, entre outras funcionalidades.

No momento da inicialização é feita uma requisição ao servidor para coletar todos os dados do atendente junto das chamadas que ainda não foram atendidas. Assim, o mesmo pode respondê-las mesmo não estando autenticado no momento da ligação.

O SAC *Manager* é composto de quatro telas e a lista de chamadas, que serão descritos a seguir.

5.2.2.1 Login

Tela de login básica (Figura 24), na qual o atendente se autentica para receber as ligações que ele já realizou e para realizar outras e ter sua conta anexada a elas. Não existe uma tela de cadastro, pois o administrador do sistema deve fazer o cadastro de atendentes manualmente.

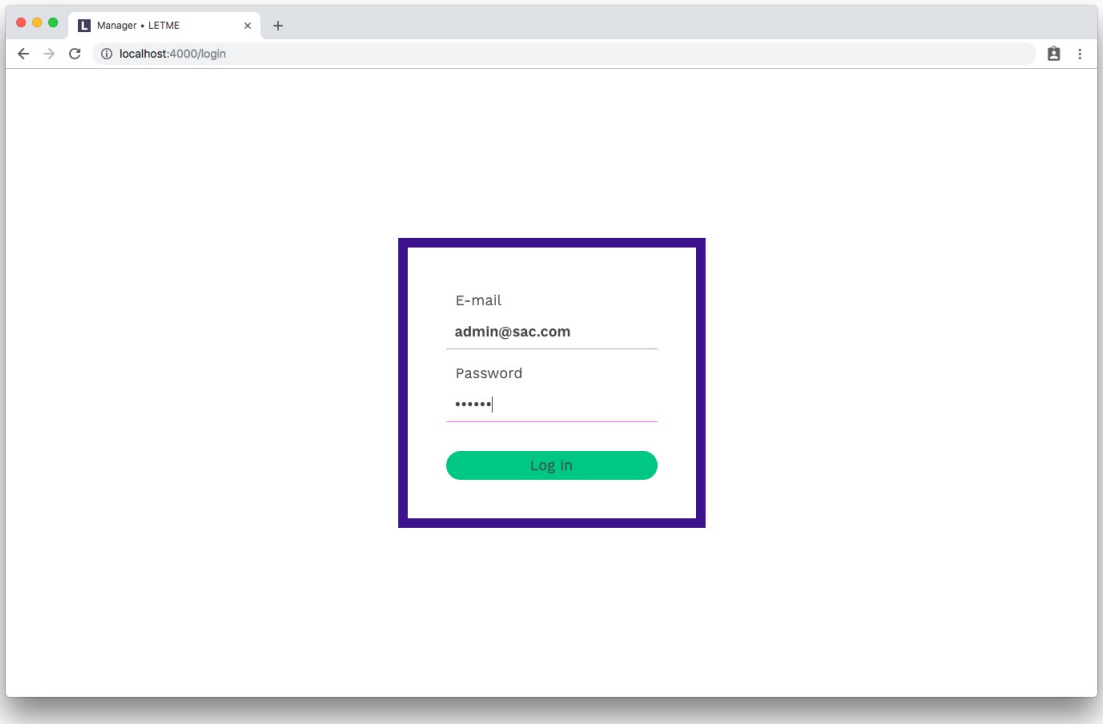

Figura 24 – Tela de login do atendente

5.2.2.2 Vídeoconferência

Principal tela do sistema, em que o atendente consegue realizar a chamada de vídeo e tentar resolver o problema do cliente. É nessa tela que a conexão *peer-to-peer* é

realizada. O vídeo é transmitido através de *streams* disponibilizados pela a API WebRTC do navegador.

Observamos na primeira tela (Figura 25) que o atendente está esperando a conexão com o cliente e o software oferece a possibilidade de desligar a ligação. Nessa tela, o atendente também tem uma visão da sua própria câmera.

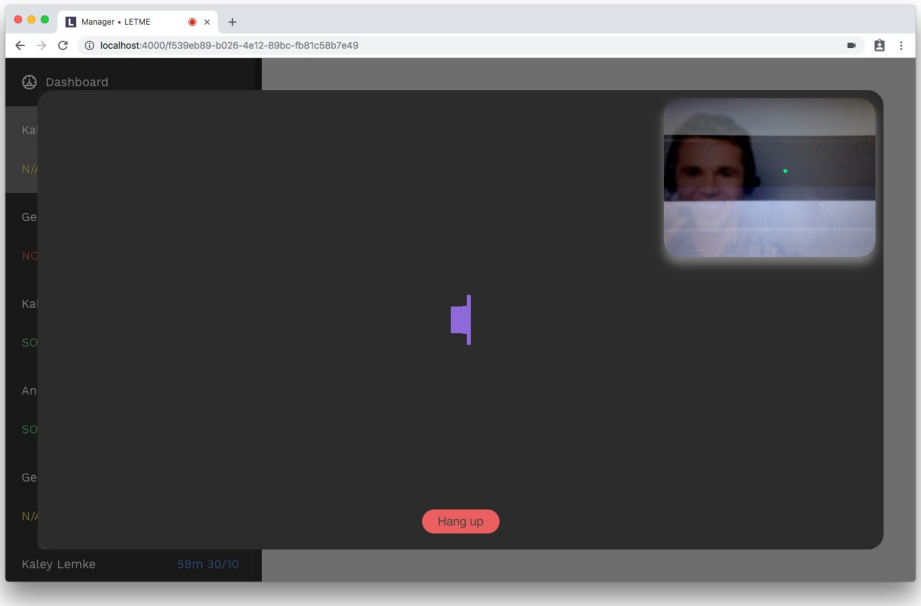

Figura 25 – Atendente esperando conexão com cliente

Na segunda tela (Figura 26) vemos o cliente já conectado ao atendente.

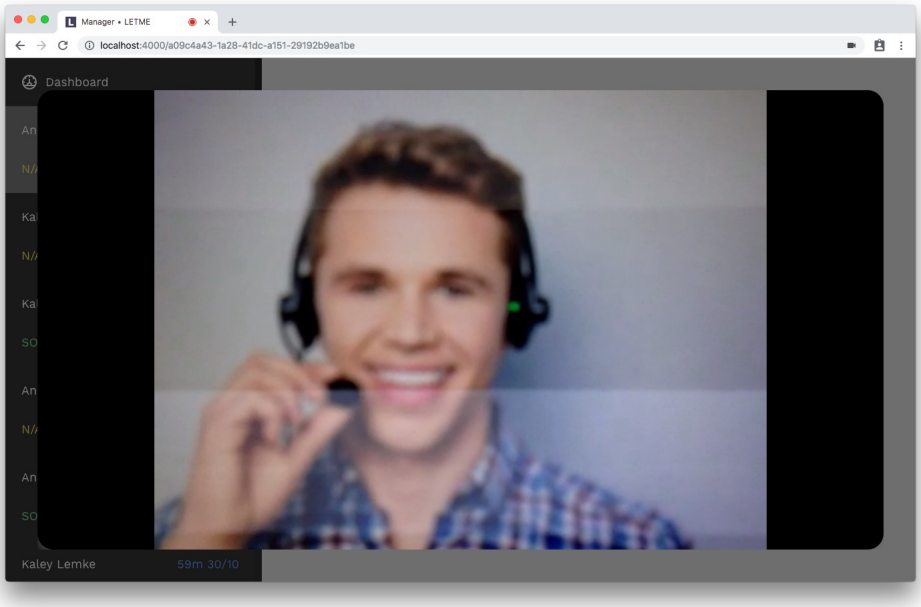

Figura 26 – Cliente aparecendo para o atendente

5.2.2.3 Estatísticas do atendente

Segunda tela mais importante, reúne informações de todas as ligações que o atendente fez, sintetizadas em um único lugar através de indicadores como média (*avg*) de avaliação e quantidade de chamadas resolvidas e não resolvidas.

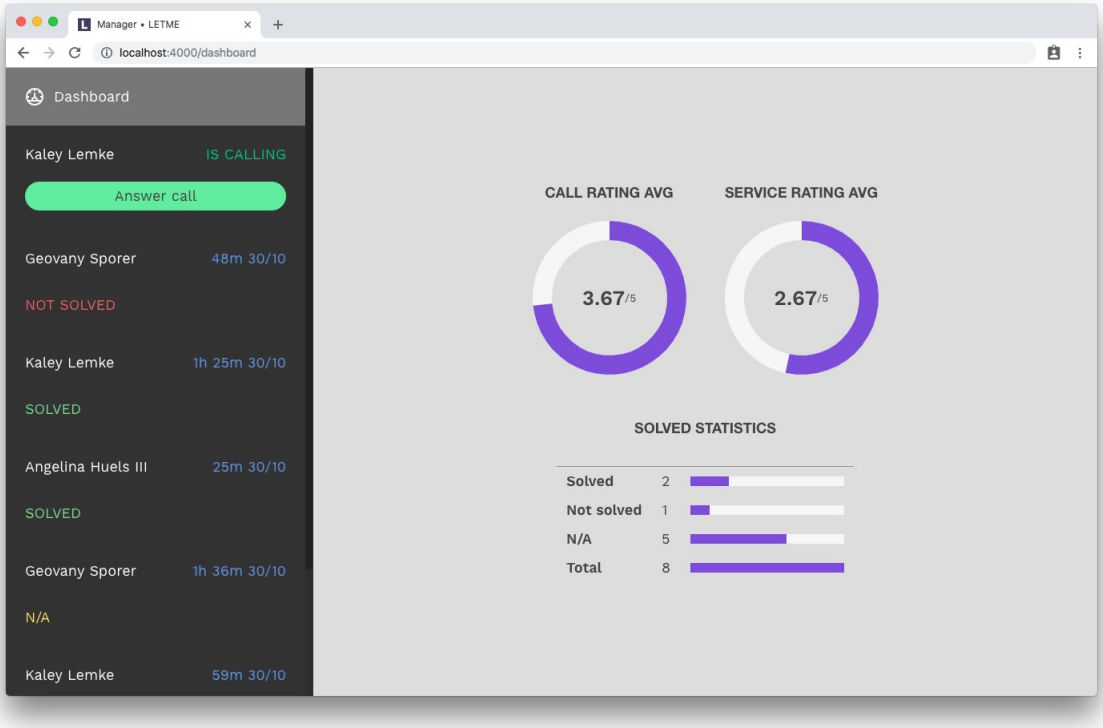

Figura 27 – Estatísticas do atendente

Vemos em Figura 27 que existem dados especificados como *N/A* (sem resposta). Isso é devido ao usuário ter a possibilidade de fechar a janela do navegador sem ter respondido o questionário de *feedback* no final da ligação.

5.2.2.4 Estatísticas da ligação

Tela utilizada para saber em mais detalhes informações sobre a chamada de serviço realizada. Nela temos outras informações como descrição da chamada que é fornecida pelo cliente e dados do mesmo, para que contatos futuros possam ser realizados se o problema não for resolvido.

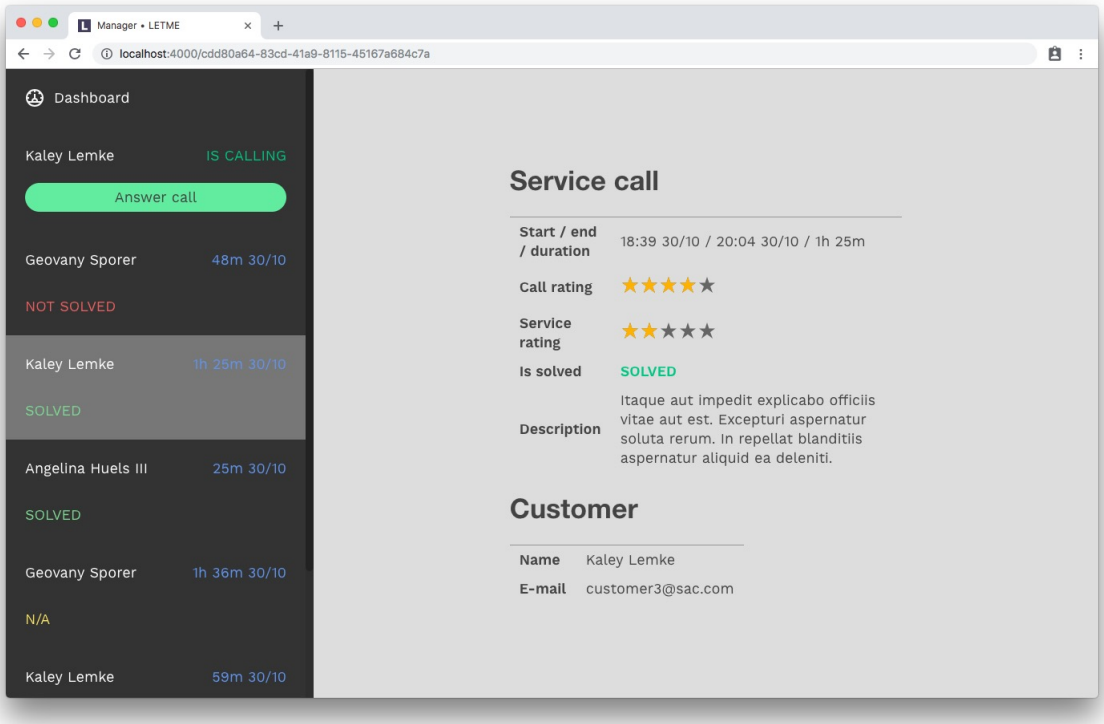

Figura 28 – Estatísticas da ligação

Observamos na imagem (Figura 28) avaliações da chamada (*call*) e do serviço (*service*), como também informações de data e duração da mesma.

5.2.2.5 Lista de chamadas

Outra parte crucial do SAC Manager é a lista de chamadas, onde temos um visão rápida sobre as chamadas que foram realizadas e também sobre clientes que estão solicitando atendimento no momento.

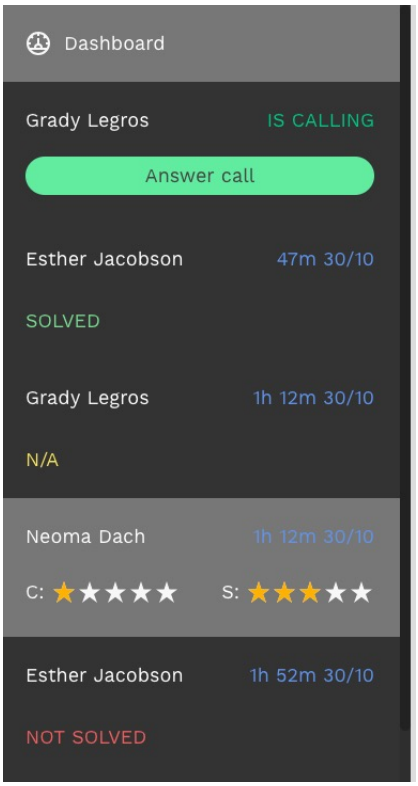

Figura 29 – Lista de chamadas de serviço

A lista (Figura 29) é atualizada em tempo real através de *web sockets*, ou seja, não é necessário atualizar a página e fazer uma requisição ao servidor. Ao mesmo tempo, quando uma chamada é atendida por outro usuário, a mesma desaparece da lista. Cada atendente somente consegue visualizar suas próprias chamadas.

5.2.3 SAC *Caller*

A aplicação do cliente, denominada SAC *Caller*, tem o propósito de ser simples e pequena para que não seja necessário o cliente baixar uma aplicação grande através de sua conexão 4G (caso Wi-Fi não esteja disponível). Essa parte do sistema conta com duas telas: a tela principal, que realiza a conexão em vídeo, e um formulário de *feedback* para o cliente avaliar a qualidade da chamada e do atendimento recebido.

O projeto SAC *Caller* foi projetado para ser exibido tanto em telas pequenas de *smartphones* quanto em telas grandes de computadores. Para demostrar essa capacidade, as figuras mostram a tela reduzida.

5.2.3.1 Videoconferência

Trata-se da tela mais importante para o cliente, na qual ele faz a requisição de chamada para o sistema, e a requisição aparece na lista de chamadas (Figura 29) dos atendentes. O funcionamento é muito similar à videoconferência do SAC *manager*.

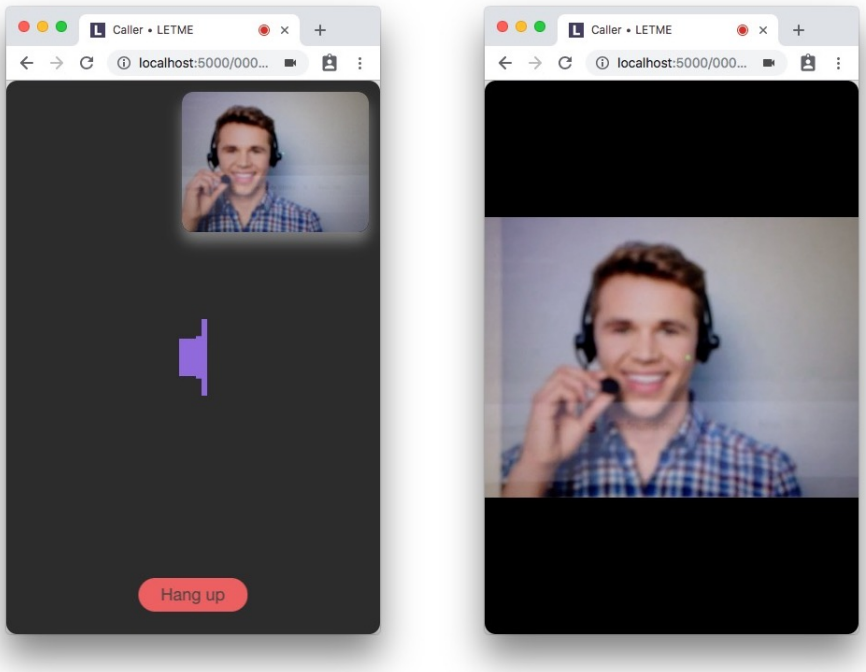

Figura 30 – Cliente em videoconferência

5.2.3.2 Formulário de qualidade

Vemos na figura (Figura 31) o formulário fornecido ao cliente após o fim da ligação.

Este formulário tem como objetivo reunir informações tanto da qualidade do atendimento recebido quanto da qualidade da chamada em si (áudio, vídeo). Por fim, o

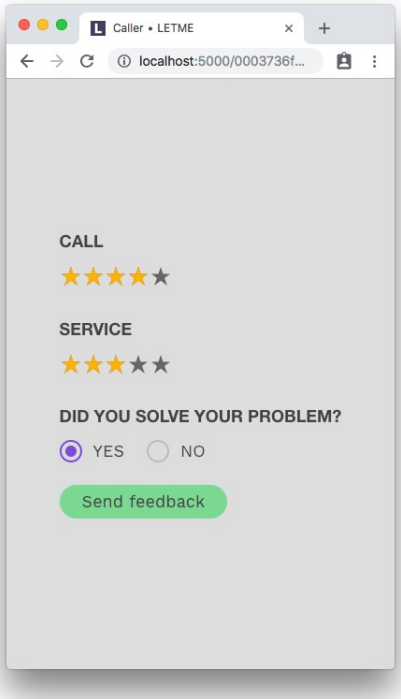

Figura 31 – Formulário de qualidade fornecido ao cliente

cliente deve informar se o problema relatado foi resolvido ou não. Em futuras atualizações do software, esse tipo de dado por ser utilizado para a empresa ter melhor conhecimento sobre a percepção dos clientes acerca de seus produtos e sobre os principais problemas relatados ao SAC.

5.2.4 SAC *Form*

Vimos nas seções anteriores os projetos e tecnologias envolvidas para realizarmos a videoconferência. Ao olharmos especificamente para o projeto SAC *Caller* (lado do cliente/consumidor, ver subseção 5.2.3), conseguimos realizar que não é uma aplicação trivial (possuí conexão através de *sockets* e *peer-to-peer*) o que resultou em um projeto relativamente grande.

Quando nos referimos a navegação na Internet e principalmente através de celulares e conexões 3G é essencial que os dados transmitidos do servidor para o cliente sejam menores possíveis (imagens comprimidas, *scripts* reduzidos através de compressão). Dentro desse contexto e sabendo que um dos objetivos do projeto, é a fácil integração do mesmo nos sites de empresas que desejam esse tipo de atendimento e a não instalação de componentes adicionais (Flash, etc) no dispositivo do cliente, foi decidido incluir o SAC *Form* na arquitetura.

Esse formulário é uma forma de incluir no site das empresas a menor quantidade

possível de código e ainda assim conseguir integrar o projeto. A razão dessa decisão foi devido a não prejudicar o uso do site por clientes que não desejam realizar o atendimento em vídeo. Economizando assim banda larga 3G e memória do site.

O funcionamento é simples, é realizada uma requisição HTTP através do formulário para cadastrar o cliente e requisitar um novo atendimento e se bem sucedida o cliente recebe um link que aponta para o projeto SAC *Caller* e inicia uma nova chamada com um atendente.

6 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo resolver o problema da impessoalidade nos serviços de atendimento devido ao aumento da tecnologia nos processos, como podemos ver no Capítulo 1.

Com o aumento de serviços disponibilizados *online* a necessidade de lojas físicas é cada vez menor, surge a seguinte dúvida: como tornar o atendimento ao consumidor mais pessoal. Essa resposta o trabalho se propôs a resolver através das chamadas de vídeo, integrando as mesmas em um sistema de atendimento ao consumidor.

Após definida a solução (conversa em vídeo) para o primeiro problema, foram elencadas algumas tecnologias que serviriam para realizar o *software*, isso levando em conta requisitos para que o atendimento não fosse muito trabalhoso para nenhuma das partes. Ambos objetivos foram alcançados e podem ser vistos na seção 1.2 do projeto.

Realizar esse trabalho proporcionou um aprendizado importante sobre como enxergar e arquitetar a partir de uma visão geral até detalhes de implementação. Foi preciso estar em contato com aplicações de gerência de informações, como banco de dados. Descobrir uma maneira de distribuir essas informações em diferentes aplicações, percorrendo áreas como protocolos de rede, conexão entre processos e arquitetura REST. E por último, foi crucial para a aplicação o conceito de transmissão de informações em tempo real, realizada com *web sockets*.

Uma aplicação com diversas partes conectadas (cliente, empresa, atendente) engloba diferentes áreas de conhecimento, em virtude disso pode-se dizer que o projeto foi multidisciplinar.

6.1 TRABALHOS FUTUROS

Existem algumas extensões do sistema que podem ser adicionadas para melhorar o resultado da ferramenta com as empresas e os seus consumidores.

Um dos motivos para incluir um formulário de *feedback* no projeto é a coleta de dados dos atendimentos. As informações entradas atualmente (qualidade do serviço e chamada) não vão muito a fundo na questão do problema, porém no futuro podem ser incluídas outras, como por exemplo: categorização do problema e do cliente; palavras-chave e descrição; entre outras.

Com esses dados um banco de informações pode ser realizado acompanhado de um *helpcenter* que serviria para o auto-atendimento de clientes e solução de problemas simples.

Um outro uso seria a análise desses dados para descobrir informações importantes sobre o produto, seu meio de produção e possíveis falhas que possam existir no processo.

O principal foco da aplicação foi realizar a chamada de vídeo através de uma conexão *peer-to-peer* e conseguir gerenciar esse fluxo chamada-resposta com diversos atendentes, portanto melhorias de design e performance são uma ótima escolha:

- Testes de usabilidade seriam um próximo passo para analisar necessidades tanto das empresas e atendentes como dos clientes; e
- Testes de performance, de conectividade e de qualidade de chamada também seriam úteis para avaliar o funcionamento a aplicação, visto que ela não foi testada em ambiente controlado com simulação de múltiplas conexões simultâneas.

REFERÊNCIAS

BERGKVIST, A. et al. *WebRTC 1.0: Real-time Communication Between Browsers*. [S.l.], 2017. Disponível em: <https://www.w3.org/TR/webrtc/>. Citado 2 vezes nas páginas 27 e 28.

BURNETT, D. C. et al. *Media Capture and Streams*. [S.l.], 2017. Disponível em: <https://www.w3.org/TR/mediacapture-streams/>. Citado na página 27.

CHAZAL, A. *The Untapped Potential Of Ecommerce Video Chat*. 2015. Online. Disponível em: <https://customericare.com/infographics-the-untapped-potential-of-ecommerce-video-chat/>. Citado na página 19.

CONSUMIDORMODERNO. *Cenário dos SACs no Brasil*. [S.l.], 2015. Página 39. Disponível em: <http://consumidormoderno.com.br/novo/wp-content/uploads/2015/12/CM208.pdf>. Citado na página 18.

DIXON, M.; FREEMAN, K.; TOMAN, N. *Stop Trying to Delight Your Customers*. 2010. Disponível em: <https://hbr.org/2010/07/stop-trying-to-delight-your-customers>. Citado na página 17.

DUTTON, S. *WebRTC in the real world: STUN, TURN and signaling*. 2013. Disponível em: <https://www.html5rocks.com/en/tutorials/webrtc/infrastructure/>. Citado 2 vezes nas páginas 11 e 29.

FARRIS, P. et al. *Marketing Metrics: The Manager's Guide to Measuring Marketing Performance*. [S.l.]: Pearson FT Press; 3 edition (September 6, 2015), 2015. Citado na página 17.

FETTE, I.; MELNIKOV, A. *The WebSocket Protocol*. [S.l.], 2011. Disponível em: <https://www.rfc-editor.org/rfc/rfc6455.txt>. Citado na página 26.

FIELDING, R. et al. *Hypertext Transfer Protocol – HTTP/1.1*. [S.l.], 1999. Disponível em: <https://www.rfc-editor.org/rfc/rfc2616.txt>. Citado na página 25.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doctoral dissertation) — University of California, Irvine, 2000. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>. Citado na página 24.

FOSSER, E.; NEDBERG, L. *Quality of Experience of WebRTC based video communication*. mathesis, 2018. Disponível em: <https://ieeexplore.ieee.org/document/8385549/>. Citado na página 39.

HUSPENI, A. Video chat: It ain't just for long distance relationships anymore (infographic). *Entrepreneur*, 2013. Disponível em: <https://www.entrepreneur.com/article/229110>. Citado na página 20.

HåKANSSON, S. *Beyond HTML5: Peer-to-Peer Conversational Video*. 2011. Disponível em: <https://www.ericsson.com/research-blog/beyond-html5-peer-peer-conversational-video/>. Citado na página 27.

INTERNATIONAL, E. *Standard ECMA-262*. [S.l.], 2017. Disponível em: <https://www.ecma-international.org/publications/standards/Ecma-262.htm>. Citado na página 49.

NEWVOICEMEDIA. *What Contact Centres are Doing Right Now*. [S.l.], 2014. Disponível em: <http://pages.newvoicemedia.com/rs/newvoicemedia/images/cch-nvm-report-what-contact-centres-are-doing-right-now-072014.pdf>. Citado na página 20.

PARMAR, M. T. H. *Adobe's Real Time Messaging Protocol*. [S.l.], 2012. Disponível em: <http://wwwimages.adobe.com/www.adobe.com/content/dam/acom/en/devnet/rtmp/pdf/rtmp_specification_1.0.pdf>. Citado na página 31.

RIPEANU, M. Peer-to-peer architecture case study: Gnutella network. *Proceedings First International Conference on Peer-to-Peer Computing*, 2001. Disponível em: <http://ieeexplore.ieee.org/abstract/document/990433/>. Citado na página 23.

SALESFORCE. *State of Service Salesforce Research*. [S.l.], 2015. Página 19. Disponível em: <https://secure.sfdcstatic.com/assets/pdf/misc/state-of-service-report-salesforce.pdf>. Citado na página 17.

SCHOLLMEIER, R. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. *Proceedings First International Conference on Peer-to-Peer Computing*, 2001. Disponível em: <http://ieeexplore.ieee.org/document/990434/>. Citado na página 23.

7 APÊNDICE A - CÓDIGO

7.1 SAC API

```
1
2  /**
3   * NOME DO ARQUIVO
4   * ./constants.js
5   */
6  import createConstants from './utils/create-constants'
7
8  export const FSA = '@@socket/FSA'
9
10 export const rtcEvents = createConstants('@@rtc/')(
11   'PEER_CONNECT',
12   'PEER_SET',
13   'STREAM_SET',
14   'SIGNAL_RECEIVE',
15   'SIGNAL_SEND'
16 )
17
18 export const serviceCallEvents = createConstants('@@service-call/
       ')(
19   'ENTITY_CREATE',
20   'ENTITY_DELETE',
21   'ENTITY_UPDATE'
22 )
23
24 export const socketEvents = createConstants()(
25   'disconnect',
26   'error'
27 )
28
29 /**
30  * NOME DO ARQUIVO
31  * ./config/index.js
32  */
33 import path from 'path'
34 import dotenv from 'dotenv-safe'
35
36 /* istanbul ignore next */
37 const requireProcessEnv = (name) => {
```

```
38   if (!process.env[name]) {
39     throw new Error('You must set the ' + name + ' environment
         variable')
40   }
41   return process.env[name]
42 }
43
44 /* istanbul ignore next */
45 if (process.env.NODE_ENV !== 'production') {
46   dotenv.load({
47     path: path.join(__dirname, '../../.env'),
48     sample: path.join(__dirname, '../../.env.example'),
49     allowEmptyValues: true
50   })
51 }
52
53 module.exports = {
54   env: process.env.NODE_ENV || 'development',
55   root: path.join(__dirname, '../../'),
56   port: process.env.SERVER_PORT,
57   ip: process.env.SERVER_HOST,
58   apiRoot: process.env.API_ROOT,
59   eventsRoot: process.env.EVENTS_ROOT,
60   masterKey: requireProcessEnv('MASTER_KEY'),
61   jwtSecret: requireProcessEnv('JWT_SECRET'),
62   knex: require('./knexfile')[process.env.NODE_ENV]
63 }
64
65 /**
66  * NOME DO ARQUIVO
67  * ./config/knexfile.js
68  */
69 const path = require('path')
70
71 /* istanbul ignore next */
72 const dotenv = require('dotenv-safe')
73 dotenv.load({
74   path: path.join(__dirname, '../../.env'),
75   sample: path.join(__dirname, '../../.env.example'),
76   allowEmptyValues: true
77 })
78
79 const commonConfig = {
80   client: 'pg',
81   connection: {
82     charset: 'utf8',
```

```
83      database: `${process.env.PG_DATABASE}_${process.env.NODE_ENV
            }`,
84      host: process.env.PG_HOST,
85      password: process.env.PG_PASSWORD,
86      port: process.env.PG_PORT,
87      user: process.env.PG_USER
88    },
89    migrations: {
90      directory: path.join(__dirname, '../services/db/migrations')
91    },
92    pool: {
93      min: 2,
94      max: 10
95    },
96    seeds: {
97      directory: path.join(__dirname, '../services/db/seeds')
98    }
99  }
100
101 module.exports = {
102   development: commonConfig,
103   test: commonConfig,
104   production: commonConfig
105 }
106
107 /**
108  * NOME DO ARQUIVO
109  * ./index.js
110  */
111 require('babel-core/register')
112
113 exports = module.exports = require('./app')
114
115 /**
116  * NOME DO ARQUIVO
117  * ./utils/objection-events.js
118  */
119 const EventEmitter = require('events')
120
121 module.exports = Model => {
122   return class EventsModel extends Model {
123     static get events () {
124       if (this.enableEvents) {
125         this.eventEmitter = this.eventEmitter || new EventEmitter
                ()
126         return this.eventEmitter
127       }
```

```
128       return false
129     }
130
131     $afterDelete (...args) {
132       return Promise.resolve(super.$afterDelete(...args))
133         .then(() => {
134           const { events } = this.constructor
135           if (events) {
136             events.emit('DELETE', { data: this.toJSON() })
137           }
138         })
139     }
140
141     $afterGet (...args) {
142       return Promise.resolve(super.$afterGet(...args))
143         .then(() => {
144           const { events } = this.constructor
145           if (events) {
146             events.emit('GET', { data: this.toJSON() })
147           }
148         })
149     }
150
151     $afterInsert (...args) {
152       return Promise.resolve(super.$afterInsert(...args))
153         .then(() => {
154           const { events } = this.constructor
155           if (events) {
156             events.emit('POST', { data: this.toJSON() })
157           }
158         })
159     }
160
161     $afterUpdate (...args) {
162       return Promise.resolve(super.$afterUpdate(...args))
163         .then(() => this.$query().returning('*'))
164         .then(instance => {
165           const { events } = this.constructor
166           if (events) {
167             events.emit('PUT', { data: instance.toJSON() })
168           }
169         })
170     }
171   }
172 }
173
174 /**
```

```
175  * NOME DO ARQUIVO
176  * ./utils/create-constants.js
177  */
178 import snakeCase from 'lodash/fp/snakeCase'
179 import toUpper from 'lodash/fp/toUpper'
180 import compose from 'lodash/fp/compose'
181
182 const upperSnakeCase = compose(
183   toUpper,
184   snakeCase
185 )
186
187 export default (prefix = '') => (...keys) =>
188   keys.reduce((obj, key) => ({ ...obj, [upperSnakeCase(key)]:
        prefix + key }), {})
189
190 /**
191  * NOME DO ARQUIVO
192  * ./common/joi-validator.js
193  */
194 import { Validator } from 'objection'
195 import Joi from 'joi'
196
197 class JoiValidator extends Validator {
198   validate ({ model, json, options: { patch } }) {
199     const schema = model.constructor.schema
200     const { error: ValidationError, value } = Joi.validate(json,
          schema, {
201       abortEarly: false,
202       noDefaults: true,
203       presence: patch ? 'optional' : 'required'
204     })
205
206     if (ValidationError) {
207       throw ValidationError
208     } else {
209       return value
210     }
211   }
212 }
213
214 export default JoiValidator
215
216 /**
217  * NOME DO ARQUIVO
218  * ./common/base-model.js
219  */
```

```
220 import { compose, Model, snakeCaseMappers } from 'objection'
221 import guid from 'objection-guid'
222 import visibility from 'objection-visibility'
223 import { timestampPlugin as timestamps } from 'objection-
    timestamps'
224
225 import JoiValidator from './joi-validator'
226 import events from '../utils/objection-events'
227
228 const enhance = compose(guid(), events, timestamps(), visibility)
229 const validator = new JoiValidator()
230
231 class BaseModel extends enhance(Model) {
232   static columnNameMappers = snakeCaseMappers()
233   static timestamp = true
234
235   static get schema () {
236     throw new Error('schema not implemented')
237   }
238
239   static createValidator () {
240     return validator
241   }
242 }
243
244 export default BaseModel
245
246 /**
247  * NOME DO ARQUIVO
248  * ./api/customers/index.js
249  */
250 import { Router } from 'express'
251 import { create, read, readAll } from './controller'
252 import { master, token } from '../../services/passport'
253
254 const router = new Router()
255
256 /**
257  * @api {get} /users Retrieve customers
258  * @apiName RetrieveCustomers
259  * @apiGroup Customer
260  * @apiPermission admin
261  * @apiSuccess {Object[]} customers List of customers.
262  * @apiError {Object} 400 Some parameters may contain invalid
    values.
263  */
264 router.get('/',
```

```
265   token({ required: true, roles: ['admin'] }),
266   readAll)
267
268 /**
269  * @api {get} /users Retrieve customer
270  * @apiName RetrieveCustomer
271  * @apiGroup Customer
272  * @apiPermission admin, user
273  * @apiSuccess {Object[]} customer Customer's data.
274  * @apiError {Object} 400 Some parameters may contain invalid
       values.
275  */
276 router.get('/:id',
277   token({ required: true }),
278   read)
279
280 /**
281  * @api {post} / Create customer
282  * @apiName CreateCustomer
283  * @apiGroup Customer
284  * @apiPermission master
285  * @apiParam {String} access_token Master access_token.
286  * @apiParam {String} email Customer's email.
287  * @apiParam {String} [name] Customer's name.
288  * @apiSuccess (201) {Object} user User's data.
289  * @apiError {Object} 400 Some parameters may contain invalid
       values.
290  * @apiError 401 Master access only.
291  * @apiSuccess {Object[]} customer Customer's data.
292  * @apiError {Object} 400 Some parameters may contain invalid
       values.
293  */
294 router.post('/',
295   master(),
296   create)
297
298 export default router
299
300 /**
301  * NOME DO ARQUIVO
302  * ./api/customers/controller.js
303  */
304 import pick from 'lodash.pick'
305 import Customer from './model'
306 import { error, notFound, success } from '../../services/response
      '
307
```

```
308 const CREATE_DATA = ['email', 'name']
309
310 export const readAll = (req, res) =>
311   Customer
312     .query()
313     .select()
314     .then(success(res, 200))
315     .catch(error(res))
316
317 export const read = ({ params }, res) =>
318   Customer
319     .query()
320     .findById(params.id)
321     .then(notFound(res))
322     .then(success(res))
323     .catch(error(res))
324
325 export const create = ({ body }, res, next) =>
326   Customer
327     .query()
328     .insert(pick(body, CREATE_DATA))
329     .then(success(res, 201))
330     .catch(error(res, next))
331
332 /**
333  * NOME DO ARQUIVO
334  * ./api/customers/model.test.js
335  */
336 import Customer from './model'
337 import { createCustomer } from '../../services/db/fixtures'
338 import { truncate } from '../../../test/helpers'
339
340 beforeEach(truncate('customers'))
341
342 describe('[model] Customer', async () => {
343   test('inserts if data is valid', async () => {
344     const customer = await createCustomer({
345       name: 'Fake name',
346       email: 'e@e.com'
347     })
348     const c = await Customer.query().first().returning('*')
349     expect(customer.email).toEqual(c.email)
350   })
351
352   test('throws error if data is invalid', async () => {
353     expect.assertions(1)
354     try {
```

```
355       await Customer.query().insert({ email: '', password: '' })
356     } catch (error) {
357       expect(error.name).toEqual('ValidationError')
358     }
359   })
360 })
361
362 /**
363  * NOME DO ARQUIVO
364  * ./api/customers/index.test.js
365  */
366 import request from 'supertest'
367 import { apiRoot, masterKey } from '../../config'
368 import { createCustomer, createCustomers, createUser } from '
    ../../services/db/fixtures'
369 import { truncate } from '../../../test/helpers'
370 import express from '../../services/express'
371 import { signSync } from '../../services/jwt'
372 import routes from '.'
373
374 const factory = express(apiRoot, routes)
375
376 beforeEach(truncate('customers'))
377
378 describe('[endpoint] /customers', () => {
379   let app
380
381   beforeEach(() => {
382     app = factory().app
383   })
384
385   describe('admin', () => {
386     let admin
387     let adminSession
388
389     beforeAll(async () => {
390       admin = await createUser({ role: 'admin' })
391       adminSession = signSync(admin.id)
392     })
393
394     describe('GET', () => {
395       test('/ 200', async () => {
396         await createCustomers([{}, {}])
397
398         const { body, status } = await request(app)
399           .get(apiRoot)
400           .query({ access_token: adminSession })
```

```
401
402         expect(status).toBe(200)
403         expect(body.length).toBe(2)
404       })
405
406       test('/:id 200', async () => {
407         const customer = await createCustomer({ email: '
                customer@sac.com' })
408         const { body, headers, status } = await request(app)
409           .get(`${apiRoot}/${customer.id}`)
410           .query({ access_token: adminSession })
411
412         expect(status).toBe(200)
413         expect(headers['content-type']).toMatch(/json/)
414         expect(Array.isArray(body)).toBe(false)
415         expect(typeof body).toBe('object')
416         expect(body.id).toBe(customer.id)
417       })
418     })
419   })
420
421   describe('master', () => {
422     describe('POST', () => {
423       test('/ 201', async () => {
424         const customer = { email: 'customer@email.com' }
425         const { body, status } = await request(app)
426           .post(apiRoot)
427           .send(customer)
428           .query({ access_token: masterKey })
429
430         expect(status).toBe(201)
431         expect(typeof body).toBe('object')
432         expect(body.email).toBe(customer.email)
433       })
434     })
435   })
436 })
437
438 /**
439  * NOME DO ARQUIVO
440  * ./api/customers/model.js
441  */
442 import Joi from 'joi'
443 import path from 'path'
444
445 import BaseModel from '../../common/base-model'
446
```

```
447 export const CustomerSchema = Joi.object({
448   id: Joi.string().uuid().optional(),
449   email: Joi.string().email(),
450   name: Joi.string().optional()
451 })
452
453 class Customer extends BaseModel {
454   static get schema () {
455     return CustomerSchema
456   }
457
458   static tableName = 'customers'
459
460   static relationMappings = {
461     serviceCalls: {
462       modelClass: path.resolve(__dirname, '../service-calls/model
          '),
463       relation: BaseModel.HasManyRelation,
464       join: {
465         from: 'customers.id',
466         to: 'service_calls.customerId'
467       }
468     }
469   }
470 }
471
472 export default Customer
473
474 /**
475  * NOME DO ARQUIVO
476  * ./api/auth/index.js
477  */
478 import { Router } from 'express'
479 import { login } from './controller'
480 import { password } from '../../services/passport'
481
482 const router = new Router()
483
484 /**
485  * @api {post} /auth Authenticate
486  * @apiName Authenticate
487  * @apiGroup Auth
488  * @apiHeader {String} Authorization Basic authorization with
      email and password.
489  * @apiSuccess (Success 201) {String} token User ‘access_token‘
      to be passed to other requests.
490  * @apiSuccess (Success 201) {Object} user Current user’s data.
```

```
491  * @apiError 401 Invalid credentials.
492  */
493 router.post('/',
494   password(),
495   login)
496
497 export default router
498
499 /**
500  * NOME DO ARQUIVO
501  * ./api/auth/controller.js
502  */
503 import { sign } from '../../services/jwt'
504 import { success } from '../../services/response/'
505
506 export const login = ({ user }, res, next) =>
507   sign(user.id)
508     .then((token) => ({ token, user: user.toJSON() }))
509     .then(success(res, 201))
510     .catch(next)
511
512 /**
513  * NOME DO ARQUIVO
514  * ./api/auth/index.test.js
515  */
516 import request from 'supertest'
517 import { apiRoot } from '../../config'
518 import { verify } from '../../services/jwt'
519 import express from '../../services/express'
520 import { createUser } from '../../services/db/fixtures'
521 import { truncate } from '../../../test/helpers'
522 import routes from '.'
523
524 const factory = express(apiRoot, routes)
525
526 beforeEach(truncate('users'))
527
528 test('POST /auth 201', async () => {
529   const email = 'user@sac.com'
530   const user = await createUser({ email })
531   const { status, body } = await request(factory().app)
532     .post(apiRoot)
533     .auth(email, '123456')
534   expect(status).toBe(201)
535   expect(typeof body).toBe('object')
536   expect(typeof body.token).toBe('string')
537   expect(typeof body.user).toBe('object')
```

```
538   expect(body.user.id).toBe(user.id)
539   expect(await verify(body.token)).toBeTruthy()
540 })
541
542 test('POST /auth 400 - invalid email', async () => {
543   const { status, body } = await request(factory().app)
544     .post(apiRoot)
545     .auth('invalid', '123456')
546   expect(status).toBe(400)
547   expect(typeof body).toBe('object')
548   expect(body.error).toBeTruthy()
549   expect(body.message).toEqual(expect.arrayContaining([expect.
        stringMatching('email')]))
550 })
551
552 test('POST /auth 400 - invalid password', async () => {
553   const { status, body } = await request(factory().app)
554     .post(apiRoot)
555     .auth('user@sac.com', '123')
556   expect(status).toBe(400)
557   expect(typeof body).toBe('object')
558   expect(body.error).toBeTruthy()
559   expect(body.message).toEqual(expect.arrayContaining([expect.
        stringMatching('password')]))
560 })
561
562 test('POST /auth 401 - user does not exist', async () => {
563   const { status } = await request(factory().app)
564     .post(apiRoot)
565     .auth('notuser@sac.com', '123456')
566   expect(status).toBe(401)
567 })
568
569 test('POST /auth 401 - wrong password', async () => {
570   await createUser()
571   const { status } = await request(factory().app)
572     .post(apiRoot)
573     .auth('user@sac.com', '654321')
574   expect(status).toBe(401)
575 })
576
577 test('POST /auth 401 - missing auth', async () => {
578   const { status } = await request(factory().app)
579     .post(apiRoot)
580   expect(status).toBe(401)
581 })
582
```

```
583 /**
584  * NOME DO ARQUIVO
585  * ./api/service-calls/events.js
586  */
587 import { FSA } from '../../constants'
588 import ServiceCall from './model'
589
590 export default io => {
591   ServiceCall.events.on('POST', ({ data }) => {
592     io.of('/manager').emit(FSA, {
593       type: '@@service-call/ENTITY_CREATE',
594       payload: { entity: data }
595     })
596   })
597   ServiceCall.events.on('PUT', ({ data }) => {
598     const action = {
599       type: '@@service-call/ENTITY_UPDATE',
600       payload: { entity: data }
601     }
602     io.of('/manager').emit(FSA, action)
603     io.of('/caller')
604       .to(data.id)
605       .emit(FSA, action)
606   })
607   ServiceCall.events.on('DELETE', ({ data }) => {
608     io.of('/manager').emit(FSA, {
609       type: '@@service-call/ENTITY_DELETE',
610       payload: { entity: data }
611     })
612   })
613 }
614
615 /**
616  * NOME DO ARQUIVO
617  * ./api/service-calls/index.js
618  */
619 import { Router } from 'express'
620 import { readAll } from './controller'
621 import { token } from '../../services/passport'
622
623 const router = new Router()
624
625 /**
626  * @api {get} /users Retrieve service calls
627  * @apiName RetrieveServiceCalls
628  * @apiGroup ServiceCalls
629  * @apiPermission admin, user
```

```
630  * @apiSuccess {Object[]} serviceCalls List of service calls.
631  * @apiError {Object} 400 Some parameters may contain invalid
         values.
632  */
633 router.get('/',
634   token({ required: true }),
635   readAll)
636
637 export default router
638
639 /**
640  * NOME DO ARQUIVO
641  * ./api/service-calls/controller.js
642  */
643 import { buildFilter } from 'objection-filter'
644 import ServiceCall from './model'
645 import { error, success } from '../../services/response'
646
647 export const readAll = ({ user, query }, res) =>
648   buildFilter(ServiceCall)
649     .build(JSON.parse(query.filter))
650     .where('userId', user.id)
651     .orWhere('userId', null)
652     .then(success(res, 200))
653     .catch(error(res))
654
655 /**
656  * NOME DO ARQUIVO
657  * ./api/service-calls/model.test.js
658  */
659 import ServiceCall from './model'
660 import { createServiceCall } from '../../services/db/fixtures'
661 import { truncate } from '../../../test/helpers'
662
663 beforeEach(truncate('service_calls'))
664
665 describe('[model] ServiceCall', async () => {
666   test('inserts if data is valid', async () => {
667     const uuid = 'b47a1f3f-9f28-4e28-8c01-c8badcdaab3e'
668     const serviceCall = await createServiceCall({ id: uuid })
669     const sc = await ServiceCall.query().first().returning('*')
670     expect(serviceCall.id).toEqual(sc.id)
671   })
672
673   test('throws error if data is invalid', async () => {
674     expect.assertions(1)
675     try {
```

```
676       await ServiceCall.query().insert({ duration: 180000,
              callRatings: '1' })
677     } catch (error) {
678       expect(error.name).toEqual('ValidationError')
679     }
680   })
681 })
682
683 /**
684  * NOME DO ARQUIVO
685  * ./api/service-calls/model.js
686  */
687 import JoiBase from 'joi'
688 import JoiDate from 'joi-date-extensions'
689 import path from 'path'
690
691 import BaseModel from '../../common/base-model'
692
693 const Joi = JoiBase.extend(JoiDate)
694
695 export const ServiceCallSchema = Joi.object({
696   id: Joi.string().uuid().optional(),
697   description: Joi.string().optional(),
698
699   // call timeline
700   startedAt: Joi.date().optional(),
701   endedAt: Joi.date().optional(),
702   duration: Joi.number().integer().forbidden(),
703
704   // call ratings
705   callRating: Joi.number().integer().min(0).max(6).optional(),
706   serviceRating: Joi.number().integer().min(0).max(6).optional(),
707   isSolved: Joi.boolean().optional(),
708
709   // relations
710   customerId: Joi.string().uuid().optional(),
711   userId: Joi.string().uuid().optional()
712 })
713
714 class ServiceCall extends BaseModel {
715   static get schema () {
716     return ServiceCallSchema
717   }
718
719   static enableEvents = true
720   static tableName = 'service_calls'
721
```

```
722   static relationMappings = {
723     customer: {
724       modelClass: path.resolve(__dirname, '../customers/model'),
725       relation: BaseModel.BelongsToOneRelation,
726       join: {
727         from: 'service_calls.customerId',
728         to: 'customers.id'
729       }
730     },
731     user: {
732       modelClass: path.resolve(__dirname, '../users/model'),
733       relation: BaseModel.BelongsToOneRelation,
734       join: {
735         from: 'service_calls.userId',
736         to: 'users.id'
737       }
738     }
739   }
740 }
741
742 export default ServiceCall
743
744 /**
745  * NOME DO ARQUIVO
746  * ./api/index.js
747  */
748 import { Router } from 'express'
749 import auth from './auth'
750 import customers from './customers'
751 import serviceCalls from './service-calls'
752 import users from './users'
753
754 const router = new Router()
755 /**
756  * // curl -v -H "Authorization: Bearer 123456789" http
        ://127.0.0.1:3000/
757  * // curl -v http://127.0.0.1:3000/?access_token=123456789
758  */
759 /**
760  * @apiDefine master Master access only
761  * You must pass ‘access_token‘ parameter or a Bearer Token
        authorization header
762  * to access this endpoint.
763  */
764 /**
765  * @apiDefine admin Admin access only
```

```
766  * You must pass `access_token` parameter or a Bearer Token
         authorization header
767  * to access this endpoint.
768  */
769 /**
770  * @apiDefine user User access only
771  * You must pass `access_token` parameter or a Bearer Token
         authorization header
772  * to access this endpoint.
773  */
774 /**
775  * @apiDefine listParams
776  * @apiParam {String} [q] Query to search.
777  * @apiParam {Number{1..30}} [page=1] Page number.
778  * @apiParam {Number{1..100}} [limit=30] Amount of returned items
         .
779  * @apiParam {String[]} [sort=-createdAt] Order of returned items
         .
780  * @apiParam {String[]} [fields] Fields to be returned.
781  */
782 router.use('/auth', auth)
783 router.use('/customers', customers)
784 router.use('/service-calls', serviceCalls)
785 router.use('/users', users)
786
787 export default router
788
789 /**
790  * NOME DO ARQUIVO
791  * ./api/users/index.js
792  */
793 import { Router } from 'express'
794 import { create, destroy, read, readAll, readMe, update,
        updatePassword } from './controller'
795 import { master, token } from '../../services/passport'
796
797 const router = new Router()
798
799 /**
800  * @api {get} /users Retrieve users
801  * @apiName RetrieveUsers
802  * @apiGroup User
803  * @apiPermission admin
804  * @apiParam {String} access_token User access_token.
805  * @apiSuccess {Object[]} users List of users.
806  * @apiError {Object} 400 Some parameters may contain invalid
         values.
```

```
807  */
808 router.get('/',
809   token({ required: true, roles: ['admin'] }),
810   readAll)
811
812 /**
813  * @api {get} /users/me Retrieve current user
814  * @apiName RetrieveCurrentUser
815  * @apiGroup User
816  * @apiPermission admin, user
817  * @apiParam {String} access_token User access_token.
818  * @apiSuccess {Object} user User's data.
819  */
820 router.get('/me',
821   token({ required: true }),
822   readMe)
823
824 /**
825  * @api {get} /users/:id Retrieve user
826  * @apiName RetrieveUser
827  * @apiGroup User
828  * @apiPermission public
829  * @apiSuccess {Object} user User's data.
830  * @apiError 404 User not found.
831  */
832 router.get('/:id', read)
833
834 /**
835 * @api {post} /users Create user
836 * @apiName CreateUser
837 * @apiGroup User
838 * @apiPermission master
839 * @apiParam {String} access_token Master access_token.
840 * @apiParam {String} email User's email.
841 * @apiParam {String} [name] User's name.
842 * @apiParam {String{6..}} password User's password.
843 * @apiParam {String="user","admin"} [role=user] User's role.
844 * @apiSuccess (201) {Object} user User's data.
845 * @apiError {Object} 400 Some parameters may contain invalid
      values.
846 * @apiError 401 Master access only.
847 * @apiError 409 Email already registered.
848 */
849 router.post('/',
850   master(),
851   create)
852
```

```
853 /**
854  * @api {patch} /users/:id Update user
855  * @apiName UpdateUser
856  * @apiGroup User
857  * @apiPermission user
858  * @apiParam {String} access_token User access_token.
859  * @apiParam {String} [name] User's name.
860  * @apiSuccess (200) {Object} user User's data.
861  * @apiError {Object} 400 Some parameters may contain invalid
       values.
862  * @apiError 401 Current user or admin access only.
863  * @apiError 404 User not found.
864 */
865 router.patch('/:id',
866   token({ required: true, roles: ['admin', 'user'] }),
867   update)
868
869 /**
870  * @api {put} /users/:id/password Update password
871  * @apiName UpdatePassword
872  * @apiGroup User
873  * @apiPermission admin, user
874  * @apiParam {String{6..}} password User's new password.
875  * @apiSuccess (Success 201) {Object} user User's data.
876  * @apiError {Object} 400 Some parameters may contain invalid
       values.
877  * @apiError 401 Current user access only.
878  * @apiError 404 User not found.
879  */
880 router.put('/:id/password',
881   token({ required: true }),
882   updatePassword)
883
884 /**
885  * @api {delete} /users/:id Delete user
886  * @apiName DeleteUser
887  * @apiGroup User
888  * @apiPermission admin
889  * @apiParam {String} access_token User access_token.
890  * @apiSuccess (Success 204) 204 No Content.
891  * @apiError 401 Admin access only.
892  * @apiError 404 User not found.
893  */
894 router.delete('/:id',
895   token({ required: true, roles: ['admin'] }),
896   destroy)
897
```

```
898 export default router
899
900 /**
901  * NOME DO ARQUIVO
902  * ./api/users/controller.js
903  */
904 import pick from 'lodash.pick'
905 import User from './model'
906 import { error, notFound, success } from '../../services/response
    '
907
908 const CREATE_DATA = ['email', 'name', 'password', 'role']
909 const UPDATE_DATA = ['email', 'name', 'role']
910
911 export const create = ({ body }, res, next) =>
912   User
913     .query()
914     .insert(pick(body, CREATE_DATA))
915     .then(success(res, 201))
916     .catch(error(res, next))
917
918 export const read = ({ params }, res, next) =>
919   User
920     .query()
921     .findById(params.id)
922     .then(notFound(res))
923     .then(success(res))
924     .catch(error(res))
925
926 export const readAll = (req, res, next) =>
927   User
928     .query()
929     .select()
930     .then(success(res))
931     .catch(error(res))
932
933 export const readMe = ({ user }, res, next) =>
934   success(res)(user)
935
936 export const update = ({ body, params, user }, res, next) => {
937   const userId = params.id === 'me' ? user.id : params.id
938   const isAdmin = user.role === 'admin'
939   const isCurrentUser = userId === user.id
940
941   if (isAdmin || isCurrentUser) {
942     return User
943       .query()
```

```
944         .patch(pick(body, UPDATE_DATA))
945         .where('id', userId)
946         .returning('*')
947         .first()
948         .then(notFound(res))
949         .then(success(res))
950         .catch(error(res))
951     }
952     return res.status(401).json({
953       message: 'You can\'t change other user\'s data',
954       valid: false
955     })
956 }
957
958 export const updatePassword = ({ body, params, user }, res, next)
      => {
959     const userId = params.id === 'me' ? user.id : params.id
960     const isCurrentUser = userId === user.id
961
962     if (isCurrentUser) {
963       return User
964         .query()
965         .patch(pick(body, ['password']))
966         .where('id', userId)
967         .returning('*')
968         .first()
969         .then(notFound(res))
970         .then(success(res))
971         .catch(error(res))
972     }
973     return res.status(401).json({
974       valid: false,
975       param: 'password',
976       message: 'You can\'t change other user\'s password'
977     })
978 }
979
980 export const destroy = ({ params }, res, next) =>
981     User
982       .query()
983       .delete()
984       .where('id', params.id)
985       .returning('*')
986       .first()
987       .then(notFound(res))
988       .then(success(res))
989       .catch(error(res))
```

```
990
991 /**
992  * NOME DO ARQUIVO
993  * ./api/users/model.test.js
994  */
995 import User from './model'
996 import { createUser } from '../../services/db/fixtures'
997 import { truncate } from '../../../test/helpers'
998
999 let data = {
1000   name: 'Fake name',
1001   email: 'e@e.com',
1002   password: '123456',
1003   role: 'admin'
1004 }
1005
1006 beforeEach(truncate('users'))
1007
1008 describe('[model] User', async () => {
1009   let user
1010
1011   beforeAll(async () => {
1012     user = await createUser(data)
1013   })
1014
1015   describe('auth', () => {
1016     test('hashes password automatically', () => {
1017       expect(user.password).not.toBe(data.password)
1018     })
1019
1020     test('verify returns true if password is correct', async ()
          => {
1021       expect(await user.verifyPassword(data.password)).toBe(true)
1022     })
1023
1024     test('verify returns false if password is wrong', async () =>
          {
1025       expect(await user.verifyPassword('123')).toBe(false)
1026     })
1027   })
1028
1029   test('throws error if data is invalid', async () => {
1030     expect.assertions(1)
1031     try {
1032       await User.query().insert({ email: '', password: '' })
1033     } catch (error) {
1034       expect(error.name).toEqual('ValidationError')
```

```
1035      }
1036    })
1037  })
1038
1039  /**
1040   * NOME DO ARQUIVO
1041   * ./api/users/index.test.js
1042   */
1043  import request from 'supertest'
1044  import { apiRoot, masterKey } from '../../config'
1045  import { createUser, createUsers } from '../../services/db/
          fixtures'
1046  import { truncate } from '../../../test/helpers'
1047  import express from '../../services/express'
1048  import { signSync } from '../../services/jwt'
1049  import User from './model'
1050  import routes from '.'
1051
1052  const factory = express(apiRoot, routes)
1053
1054  beforeEach(truncate('users'))
1055
1056  describe('[endpoint] /users', () => {
1057    let app
1058
1059    beforeEach(() => {
1060      app = factory().app
1061    })
1062
1063    describe('admin', () => {
1064      describe('DELETE', () => {
1065        test('/:id 200', async () => {
1066          const email = 'user@email.com'
1067          const [admin, user] = await createUsers([
1068            {
1069              name: 'Admin',
1070              email: 'useradmin@email.com',
1071              role: 'admin'
1072            },
1073            {
1074              email
1075            }
1076          ])
1077          const adminSession = signSync(admin.id)
1078          const { status, body } = await request(app)
1079            .delete(`${apiRoot}/${user.id}`)
1080            .send({ access_token: adminSession })
```

```
1081
1082        expect(status).toBe(200)
1083        expect(body.email).toEqual(email)
1084      })
1085
1086      test('/:id 404', async () => {
1087        const admin = await createUser({ role: 'admin' })
1088        const adminSession = signSync(admin.id)
1089        const { status } = await request(app)
1090          .delete(apiRoot + '/5a21b571-c60f-4d1c-abfd-23
                e06900cb2e')
1091          .send({ access_token: adminSession })
1092
1093        expect(status).toBe(404)
1094      })
1095    })
1096
1097    describe('GET', () => {
1098      test('/ 200', async () => {
1099        const [admin] = await createUsers([
1100          {
1101            name: 'Admin',
1102            email: 'useradmin@email.com',
1103            role: 'admin'
1104          },
1105          {},
1106          {}
1107        ])
1108        const adminSession = signSync(admin.id)
1109        const { body, headers, status } = await request(app)
1110          .get(apiRoot)
1111          .query({ access_token: adminSession })
1112
1113        expect(status).toBe(200)
1114        expect(headers['content-type']).toMatch(/json/)
1115        expect(Array.isArray(body)).toBe(true)
1116        expect(body.length).toBe(3)
1117
1118        expect(body[0]).toEqual(expect.objectContaining({
1119          email: expect.any(String),
1120          name: expect.any(String),
1121          role: expect.any(String),
1122          password: expect.any(String)
1123        }))
1124      })
1125    })
1126
```

```
1127      describe('PATCH', () => {
1128        test('/:id 200', async () => {
1129          const admin = await createUser({ role: 'admin' })
1130          const user = await createUser({ name: 'othername' })
1131          const adminSession = signSync(admin.id)
1132
1133          const { status, body } = await request(app)
1134            .patch(`${apiRoot}/${user.id}`)
1135            .send({ access_token: adminSession, name: 'test' })
1136
1137          expect(status).toBe(200)
1138          expect(typeof body).toBe('object')
1139          expect(body.name).toBe('test')
1140        })
1141
1142        test('/:id 404', async () => {
1143          const admin = await createUser({ role: 'admin' })
1144          const adminSession = signSync(admin.id)
1145          const { status } = await request(app)
1146            .patch(apiRoot + '/5a21b571-c60f-4d1c-abfd-23e06900cb2e
                 ')
1147            .send({ access_token: adminSession, name: 'test' })
1148
1149          expect(status).toBe(404)
1150        })
1151      })
1152    })
1153
1154    describe('master', () => {
1155      describe('POST', () => {
1156        test('/ 201', async () => {
1157          const email = 'user@email.com'
1158          const { status, body } = await request(app)
1159            .post(apiRoot)
1160            .send({ access_token: masterKey, email, password: '
                 123456', role: 'user' })
1161
1162          expect(status).toBe(201)
1163          expect(typeof body).toBe('object')
1164          expect(body.email).toBe(email)
1165        })
1166
1167        test('/ 409 - duplicated email', async () => {
1168          const email = 'duplicated@email.com'
1169          await createUser({ email })
1170          const { status, body } = await request(app)
1171            .post(apiRoot)
```

```
1172          .send({ access_token: masterKey, email, password: '
                  123456' })
1173        expect(status).toBe(409)
1174        expect(typeof body).toBe('object')
1175        expect(body.errors[0].param).toBe('email')
1176      })
1177
1178      test('/ 400 - invalid email', async () => {
1179        const { status, body } = await request(app)
1180          .post(apiRoot)
1181          .send({ access_token: masterKey, email: 'invalid',
                  password: '123456' })
1182        expect(status).toBe(400)
1183        expect(typeof body).toBe('object')
1184        expect(body.errors[0].param).toBe('email')
1185      })
1186
1187      test('/ 400 - missing email', async () => {
1188        const { status, body } = await request(app)
1189          .post(apiRoot)
1190          .send({ access_token: masterKey, password: '123456' })
1191        expect(status).toBe(400)
1192        expect(typeof body).toBe('object')
1193        expect(body.errors[0].param).toBe('email')
1194      })
1195
1196      test('/ 400 - invalid password', async () => {
1197        const { status, body } = await request(app)
1198          .post(apiRoot)
1199          .send({ access_token: masterKey, email: 'd@d.com',
                  password: '123' })
1200        expect(status).toBe(400)
1201        expect(typeof body).toBe('object')
1202        expect(body.errors[0].param).toBe('password')
1203      })
1204
1205      test('/ 400 - missing password', async () => {
1206        const { status, body } = await request(app)
1207          .post(apiRoot)
1208          .send({ access_token: masterKey, email: 'd@d.com' })
1209        expect(status).toBe(400)
1210        expect(typeof body).toBe('object')
1211        expect(body.errors[0].param).toBe('password')
1212      })
1213
1214      test('/ 400 - invalid role', async () => {
1215        const { status, body } = await request(app)
```

```
1216          .post(apiRoot)
1217          .send({ access_token: masterKey, email: 'd@d.com',
                password: '123456', role: 'invalid' })
1218        expect(status).toBe(400)
1219        expect(typeof body).toBe('object')
1220        expect(body.errors[0].param).toBe('role')
1221      })
1222    })
1223  })
1224
1225  describe('unauth', () => {
1226    describe('DELETE', () => {
1227      test('/:id 401', async () => {
1228        const user = await createUser()
1229        const { status } = await request(app)
1230          .delete(`${apiRoot}/${user.id}`)
1231        expect(status).toBe(401)
1232      })
1233    })
1234
1235    describe('GET', () => {
1236      test('/ 401', async () => {
1237        const { status } = await request(app)
1238          .get(apiRoot)
1239        expect(status).toBe(401)
1240      })
1241
1242      test('/me 401', async () => {
1243        const { status } = await request(app)
1244          .get(apiRoot + '/me')
1245        expect(status).toBe(401)
1246      })
1247
1248      test('/:id 200', async () => {
1249        const user = await createUser()
1250        const response = await request(app)
1251          .get(`${apiRoot}/${user.id}`)
1252        const { body, headers, status } = response
1253
1254        expect(status).toBe(200)
1255        expect(headers['content-type']).toMatch(/json/)
1256        expect(Array.isArray(body)).toBe(false)
1257        expect(typeof body).toBe('object')
1258        expect(body.id).toBe(user.id)
1259      })
1260
1261      test('/:id 404', async () => {
```

```
1262        const { status } = await request(app)
1263          .get(apiRoot + '/5a21b571-c60f-4d1c-abfd-23e06900cb2e')
1264        expect(status).toBe(404)
1265      })
1266    })
1267
1268    describe('PATCH', () => {
1269      test('/me 401', async () => {
1270        const { status } = await request(app)
1271          .patch(apiRoot + '/me')
1272          .send({ name: 'test' })
1273        expect(status).toBe(401)
1274      })
1275
1276      test('/:id 401', async () => {
1277        const user = await createUser()
1278        const { status } = await request(app)
1279          .patch(`${apiRoot}/${user.id}`)
1280          .send({ name: 'test' })
1281        expect(status).toBe(401)
1282      })
1283    })
1284  })
1285
1286  describe('user', () => {
1287    describe('DELETE', () => {
1288      test('/:id 401', async () => {
1289        const user = await createUser()
1290        const userSession = signSync(user.id)
1291        const { status } = await request(app)
1292          .delete(`${apiRoot}/${user.id}`)
1293          .send({ access_token: userSession })
1294        expect(status).toBe(401)
1295      })
1296    })
1297
1298    describe('GET', () => {
1299      test('/ 401', async () => {
1300        const user = await createUser()
1301        const userSession = signSync(user.id)
1302        const { status } = await request(app)
1303          .get(apiRoot)
1304          .query({ access_token: userSession })
1305        expect(status).toBe(401)
1306      })
1307
1308      test('/me 200', async () => {
```

```
1309        const user = await createUser()
1310        const userSession = signSync(user.id)
1311        const { status, body } = await request(app)
1312          .get(apiRoot + '/me')
1313          .query({ access_token: userSession })
1314        expect(status).toBe(200)
1315        expect(typeof body).toBe('object')
1316        expect(body.id).toBe(user.id)
1317      })
1318    })
1319
1320    describe('PATCH', () => {
1321      test('/me 200', async () => {
1322        const user = await createUser()
1323        const userSession = signSync(user.id)
1324
1325        expect(user.email).not.toBe('test@test.com')
1326        expect(user.name).not.toBe('test')
1327
1328        const { status, body } = await request(app)
1329          .patch(apiRoot + '/me')
1330          .send({ access_token: userSession, email: 'test@test.
                com', name: 'test' })
1331
1332        expect(status).toBe(200)
1333        expect(typeof body).toBe('object')
1334        expect(body.email).toBe('test@test.com')
1335        expect(body.name).toBe('test')
1336      })
1337
1338      test('/:id 200', async () => {
1339        const user = await createUser()
1340        const userSession = signSync(user.id)
1341
1342        expect(user.email).not.toBe('test@test.com')
1343        expect(user.name).not.toBe('test')
1344
1345        const { status, body } = await request(app)
1346          .patch(`${apiRoot}/${user.id}`)
1347          .send({ access_token: userSession, email: 'test@test.
                com', name: 'test' })
1348        expect(status).toBe(200)
1349        expect(typeof body).toBe('object')
1350        expect(body.email).toBe('test@test.com')
1351        expect(body.name).toBe('test')
1352      })
1353
```

```
1354        test('/:id 401 - another user', async () => {
1355          const userOne = await createUser({ name: 'userone' })
1356          const userTwo = await createUser({ name: 'usertwo' })
1357          const userOneSession = signSync(userOne.id)
1358
1359          const { status } = await request(app)
1360            .patch(`${apiRoot}/${userTwo.id}`)
1361            .send({ access_token: userOneSession, name: 'test' })
1362          expect(status).toBe(401)
1363        })
1364
1365        const verifyPassword = async (password, userId) => {
1366          const user = await User.query().findById(userId)
1367          return user.verifyPassword(password)
1368        }
1369
1370        test('/:id/password 200', async () => {
1371          const email = 'user@email.com'
1372          const user = await createUser({
1373            email,
1374            password: '123456'
1375          })
1376          const userSession = signSync(user.id)
1377          const { status, body } = await request(app)
1378            .put(`${apiRoot}/${user.id}/password`)
1379            .send({ access_token: userSession, password: '654321'
                 })
1380
1381          expect(status).toBe(200)
1382          expect(typeof body).toBe('object')
1383          expect(body.email).toBe(email)
1384          expect(await verifyPassword('654321', body.id)).toBe(true
               )
1385        })
1386
1387        test('/:id/password 400 - invalid password', async () => {
1388          const email = 'user@email.com'
1389          const user = await createUser({
1390            email,
1391            password: '123456'
1392          })
1393          const userSession = signSync(user.id)
1394          const { status, body } = await request(app)
1395            .put(`${apiRoot}/${user.id}/password`)
1396            .send({ access_token: userSession, password: '321' })
1397
1398          expect(status).toBe(400)
```

```
1399          expect(typeof body).toBe('object')
1400          expect(body.errors[0].param).toBe('password')
1401        })
1402
1403        test('/:id/password 401 - invalid authentication method',
                async () => {
1404          const email = 'user@email.com'
1405          const password = '123456'
1406          const user = await createUser({
1407            email,
1408            password
1409          })
1410          const { status } = await request(app)
1411            .put(`${apiRoot}/${user.id}/password`)
1412            .auth(email, password)
1413            .send({ password: '654321' })
1414
1415          expect(status).toBe(401)
1416        })
1417
1418        test('/:id/password 401 - another user', async () => {
1419          const userOne = await createUser()
1420          const userTwo = await createUser()
1421          const userOneSession = signSync(userOne.id)
1422          const { status } = await request(app)
1423            .put(`${apiRoot}/${userTwo.id}/password`)
1424            .send({ access_token: userOneSession, password: '654321
                  ' })
1425
1426          expect(status).toBe(401)
1427        })
1428      })
1429    })
1430  })
1431
1432  /**
1433   * NOME DO ARQUIVO
1434   * ./api/users/model.js
1435   */
1436  import Joi from 'joi'
1437  import { compose } from 'objection'
1438  import password from 'objection-password'
1439  import path from 'path'
1440
1441  import BaseModel from '../../common/base-model'
1442
1443  const enhance = compose(password())
```

```
1444
1445 export const USER_ROLE_OPTIONS = ['admin', 'user']
1446
1447 export const UserSchema = Joi.object({
1448   id: Joi.string().uuid().optional(),
1449   email: Joi.string().email(),
1450   name: Joi.string().optional(),
1451   role: Joi.string().valid(USER_ROLE_OPTIONS).optional(),
1452   password: Joi.string().min(6)
1453 })
1454
1455 class User extends enhance(BaseModel) {
1456   static get schema () {
1457     return UserSchema
1458   }
1459
1460   static tableName = 'users'
1461
1462   static relationMappings = {
1463     serviceCalls: {
1464       modelClass: path.resolve(__dirname, '../service-calls/model
          '),
1465       relation: BaseModel.HasManyRelation,
1466       join: {
1467         from: 'users.id',
1468         to: 'service_calls.userId'
1469       }
1470     }
1471   }
1472 }
1473
1474 export default User
1475
1476 /**
1477  * NOME DO ARQUIVO
1478  * ./events/caller.js
1479  */
1480 import d from 'debug'
1481 import ServiceCall from '../api/service-calls/model'
1482 import { FSA, rtcEvents, serviceCallEvents, socketEvents } from '
     ../constants'
1483
1484 const debug = d('socket:caller')
1485
1486 const sidToServiceCallId = {}
1487
```

```
1488 export const ackSuccess = ack => entity => ack && ack(null,
       entity)
1489 export const ackError = ack => error => ack && ack(error)
1490
1491 export default io => socket => {
1492   debug('socket connected')
1493
1494   // Socket domain events
1495   socket.on(socketEvents.DISCONNECT, async reason => {
1496     debug(`socket disconnected ${reason}`)
1497
1498     const svcId = sidToServiceCallId[socket.id]
1499     delete sidToServiceCallId[socket.id]
1500
1501     if (svcId) {
1502       const svc = await ServiceCall.query()
1503         .findById(svcId)
1504         .returning('*')
1505
1506       svc.$query().delete()
1507     }
1508   })
1509
1510   socket.on(socketEvents.ERROR, error => {
1511     debug(`socket error ${error}`)
1512   })
1513
1514   // Service call events
1515   socket.on(serviceCallEvents.ENTITY_CREATE, ({ data }, ack) =>
1516     ServiceCall.query()
1517       .insert(data)
1518       .returning('*')
1519       .then(entity => {
1520         sidToServiceCallId[socket.id] = entity.id
1521         ackSuccess(ack)(entity)
1522       })
1523       .catch(ackError(ack))
1524   )
1525
1526   socket.on(serviceCallEvents.ENTITY_UPDATE, ({ data }, ack) =>
1527     ServiceCall.query()
1528       .patch(data)
1529       .where('id', data.id)
1530       .returning('*')
1531       .then(entity => {
1532         ackSuccess(ack)(entity)
1533       })
```

```
1534        .catch(ackError(ack))
1535    )
1536
1537    // RTC events
1538    socket.on(rtcEvents.PEER_CONNECT, (message, ack) => {
1539      debug(`${rtcEvents.PEER_CONNECT} received`)
1540      const {
1541        meta: { room }
1542      } = message
1543
1544      socket.join(room, () => {
1545        debug(`joined ${room}`)
1546      })
1547    })
1548
1549    socket.on(rtcEvents.SIGNAL_SEND, message => {
1550      debug(`${rtcEvents.SIGNAL_SEND} received`)
1551      const {
1552        meta: { namespace, room }
1553      } = message
1554
1555      debug(`emitting ${FSA} to ${room} and ${namespace}`)
1556
1557      io.of(namespace)
1558        .to(room)
1559        .emit(FSA, message)
1560    })
1561  }
1562
1563  /**
1564   * NOME DO ARQUIVO
1565   * ./events/index.js
1566   */
1567  import callerEvents from './caller'
1568  import managerEvents from './manager'
1569  import registerServiceCallEvents from '../api/service-calls/
       events'
1570
1571  export default io => {
1572    io.of('/caller').on('connection', callerEvents(io))
1573    io.of('/manager').on('connection', managerEvents(io))
1574
1575    registerServiceCallEvents(io)
1576  }
1577
1578  /**
1579   * NOME DO ARQUIVO
```

```
1580  * ./events/manager.js
1581  */
1582 import d from 'debug'
1583 import ServiceCall from '../api/service-calls/model'
1584 import { FSA, rtcEvents, serviceCallEvents, socketEvents } from '
        ../constants'
1585
1586 const debug = d('socket:manager')
1587
1588 export const ackSuccess = ack => entity => ack && ack(null,
        entity)
1589 export const ackError = ack => error => ack && ack(error)
1590
1591 export default io => socket => {
1592   debug('socket connected')
1593
1594   // Socket domain events
1595   socket.on(socketEvents.DISCONNECT, async reason => {
1596     debug(`socket disconnected ${reason}`)
1597   })
1598
1599   socket.on(socketEvents.ERROR, error => {
1600     debug(`socket error ${error}`)
1601   })
1602
1603   // Service call events
1604   socket.on(serviceCallEvents.ENTITY_UPDATE, ({ data }, ack) =>
1605     ServiceCall
1606       .query()
1607       .patch(data)
1608       .where('id', data.id)
1609       .returning('*')
1610       .then(entity => {
1611         ackSuccess(ack)(entity)
1612       })
1613       .catch(ackError(ack))
1614   )
1615
1616   // RTC events
1617   socket.on(rtcEvents.PEER_CONNECT, (message, ack) => {
1618     debug(`${rtcEvents.PEER_CONNECT} received`)
1619     const { meta: { room } } = message
1620
1621     socket.join(room, () => {
1622       debug(`joined ${room}`)
1623     })
1624   })
```

```
1625
1626   socket.on(rtcEvents.SIGNAL_SEND, message => {
1627     debug(`${rtcEvents.SIGNAL_SEND} received`)
1628     const {
1629       meta: { namespace, room }
1630     } = message
1631
1632     debug(`emitting ${FSA} to ${room} and ${namespace}`)
1633
1634     io.of(namespace).to(room).emit(FSA, message)
1635   })
1636 }
1637
1638 /**
1639  * NOME DO ARQUIVO
1640  * ./app.js
1641  */
1642 import { Server } from 'http'
1643 import pipe from 'lodash/fp/pipe'
1644 import { env, port, ip, apiRoot, eventsRoot } from './config'
1645 import express from './services/express'
1646 import { init } from './services/db'
1647 import socketio from './services/socketio'
1648 import api from './api'
1649 import events from './events'
1650
1651 init()
1652
1653 // socket.io setup must be the last
1654 const { app, io, server } = pipe(
1655   express(apiRoot, api),
1656   socketio(eventsRoot, events)
1657 )({
1658   server: new Server()
1659 })
1660
1661 setImmediate(() => {
1662   server.listen(port, ip, () => {
1663     console.log('API and Socket Server listening on http://%s:%d,
          in %s mode', ip, port, env)
1664   })
1665 })
1666
1667 export default { app, io, server }
1668
1669 /**
1670  * NOME DO ARQUIVO
```

```
1671  * ./services/jwt/index.js
1672  */
1673 import jwt from 'jsonwebtoken'
1674 import Promise from 'bluebird'
1675 import { jwtSecret } from '../../config'
1676
1677 const jwtSign = Promise.promisify(jwt.sign)
1678 const jwtVerify = Promise.promisify(jwt.verify)
1679
1680 export const sign = (id, options, method = jwtSign) =>
1681   method({ id }, jwtSecret, options)
1682
1683 export const signSync = (id, options) => sign(id, options, jwt.
       sign)
1684
1685 export const verify = (token) => jwtVerify(token, jwtSecret)
1686
1687 /**
1688  * NOME DO ARQUIVO
1689  * ./services/response/index.js
1690  */
1691 export const success = (res, status) => (entity) => {
1692   if (entity) {
1693     res.status(status || 200).json(entity)
1694   }
1695   return null
1696 }
1697
1698 export const notFound = (res) => (entity) => {
1699   if (entity) {
1700     return entity
1701   }
1702   res.status(404).end()
1703   return null
1704 }
1705
1706 export const error = (res) => (error) => {
1707   if (error.name === 'error' && error.code === '23505') { //
       database validation error
1708     return res.status(409).json({
1709       valid: false,
1710       errors: [
1711         {
1712           param: 'email',
1713           message: '"email" already registered'
1714         }
1715       ]
```

```
1716      })
1717    } else if (error.name === 'ValidationError' && error.isJoi) {
1718      const errors = error.details.map(e => ({
1719        param: e.context.key,
1720        message: e.message
1721      }))
1722
1723      res.status(400).json({
1724        valid: false,
1725        errors
1726      })
1727    }
1728    return null
1729  }
1730
1731  /**
1732   * NOME DO ARQUIVO
1733   * ./services/response/index.test.js
1734   */
1735  import * as response from '.'
1736
1737  let res
1738
1739  beforeEach(() => {
1740    res = {
1741      status: jest.fn(() => res),
1742      json: jest.fn(() => res),
1743      end: jest.fn(() => res)
1744    }
1745  })
1746
1747  describe('success', () => {
1748    it('responds with passed object and status 200', () => {
1749      expect(response.success(res)({ prop: 'value' })).toBeNull()
1750      expect(res.status).toBeCalledWith(200)
1751      expect(res.json).toBeCalledWith({ prop: 'value' })
1752    })
1753
1754    it('responds with passed object and status 201', () => {
1755      expect(response.success(res, 201)({ prop: 'value' })).
           toBeNull()
1756      expect(res.status).toBeCalledWith(201)
1757      expect(res.json).toBeCalledWith({ prop: 'value' })
1758    })
1759
1760    it('does not send any response when object has not been passed'
         , () => {
```

```
1761      expect(response.success(res, 201)()).toBeNull()
1762      expect(res.status).not.toBeCalled()
1763    })
1764  })
1765
1766  describe('notFound', () => {
1767    it('responds with status 404 when object has not been passed',
          () => {
1768      expect(response.notFound(res)()).toBeNull()
1769      expect(res.status).toBeCalledWith(404)
1770      expect(res.end).toHaveBeenCalledTimes(1)
1771    })
1772
1773    it('returns the passed object and does not send any response',
          () => {
1774      expect(response.notFound(res)({ prop: 'value' })).toEqual({
            prop: 'value' })
1775      expect(res.status).not.toBeCalled()
1776      expect(res.end).not.toBeCalled()
1777    })
1778  })
1779
1780  /**
1781   * NOME DO ARQUIVO
1782   * ./services/express/index.js
1783   */
1784  import express from 'express'
1785  import cors from 'cors'
1786  import compression from 'compression'
1787  import morgan from 'morgan'
1788  import bodyParser from 'body-parser'
1789  import { env } from '../../config'
1790
1791  export default (apiRoot, routes) => (composed = {}) => {
1792    const { server } = composed
1793    const app = express()
1794
1795    if (env === 'production' || env === 'development') {
1796      app.use(cors())
1797      app.use(compression())
1798      app.use(morgan('dev'))
1799    }
1800
1801    app.use(bodyParser.urlencoded({ extended: false }))
1802    app.use(bodyParser.json())
1803    app.use(apiRoot, routes)
1804
```

```
1805   // error handlers
1806
1807   // development error handler
1808   // will print stacktrace
1809   if (env === 'development') {
1810     app.use(function (error, req, res, next) {
1811       res
1812         .status(error.status || 500)
1813         .json({
1814           message: error.message,
1815           error
1816         })
1817     })
1818   }
1819
1820   // production error handler
1821   // no stacktraces leaked to user
1822   app.use((error, req, res, next) => {
1823     res
1824       .status(error.status || 500)
1825       .json({
1826         message: error.message,
1827         error: {}
1828       })
1829   })
1830
1831   composed.app = app
1832   if (server) {
1833     composed.server = server.on('request', app)
1834   }
1835
1836   return composed
1837 }
1838
1839 /**
1840  * NOME DO ARQUIVO
1841  * ./services/passport/index.js
1842  */
1843 import passport from 'passport'
1844 import { BasicStrategy } from 'passport-http'
1845 import { Strategy as BearerStrategy } from 'passport-http-bearer'
1846 import { Strategy as JwtStrategy, ExtractJwt } from 'passport-jwt
      '
1847 import { jwtSecret, masterKey } from '../../config'
1848 import User, { UserSchema, USER_ROLE_OPTIONS } from '../../api/
      users/model'
1849
```

```
1850 const formatError = (error) =>
1851   error.isJoi ? {
1852     message: error.details.map(detail => detail.message),
1853     error
1854   } : error
1855
1856 export const password = () => (req, res, next) =>
1857   passport.authenticate('password', { session: false }, (error,
         user, info) => {
1858     if (error) {
1859       return res.status(400).json(formatError(error))
1860     } else if (!user) {
1861       return res.status(401).end()
1862     }
1863     req.login(user, { session: false }, (err) => {
1864       if (err) return res.status(401).end()
1865       next()
1866     })
1867   })(req, res, next)
1868
1869 export const master = () =>
1870   passport.authenticate('master', { session: false })
1871
1872 export const token = ({ required, roles = USER_ROLE_OPTIONS } =
       {}) => (req, res, next) =>
1873   passport.authenticate('token', { session: false }, (err, user,
         info) => {
1874     if (err) return next(err)
1875     if ((required && !user) || (required && !roles.includes(user.
           role))) {
1876       return res.status(401).end()
1877     }
1878     req.login(user, { session: false }, (err) => {
1879       if (err) return res.status(401).end()
1880       next()
1881     })
1882   })(req, res, next)
1883
1884 passport.use('password', new BasicStrategy(async (email, password
       , done) => {
1885   try {
1886     await UserSchema.validate({ email, password }, { abortEarly:
           false })
1887   } catch (error) {
1888     return done(error)
1889   }
1890
```

```
1891    const user = await User
1892      .query()
1893      .where('email', email)
1894      .first()
1895
1896    if (!user) {
1897      return done(null, false, { message: 'No user was found.' })
1898    }
1899
1900    const isVerified = await user.verifyPassword(password)
1901
1902    if (isVerified) {
1903      return done(null, user)
1904    } else {
1905      return done(null, false)
1906    }
1907  }))
1908
1909  passport.use('master', new BearerStrategy((token, done) => {
1910    if (token === masterKey) {
1911      return done(null, {}, { role: 'master' })
1912    } else {
1913      return done(null, false)
1914    }
1915  }))
1916
1917  passport.use('token', new JwtStrategy({
1918    secretOrKey: jwtSecret,
1919    jwtFromRequest: ExtractJwt.fromExtractors([
1920      ExtractJwt.fromUrlQueryParameter('access_token'),
1921      ExtractJwt.fromBodyField('access_token'),
1922      ExtractJwt.fromAuthHeaderAsBearerToken()
1923    ])
1924  }, ({ id }, done) =>
1925    User
1926      .query()
1927      .findById(id)
1928      .asCallback((err, user) => {
1929        if (err) {
1930          return done(err)
1931        }
1932        return done(null, user)
1933      })
1934  ))
1935
1936  /**
1937   * NOME DO ARQUIVO
```

```
1938  * ./services/socketio/index.js
1939  */
1940 import socketio from 'socket.io'
1941
1942 export default (eventsRoot = '', events = f => f) => (composed =
        {}) => {
1943   const { server } = composed
1944   const io = socketio({
1945     path: eventsRoot,
1946     serveClient: false
1947   })
1948   events(io)
1949   composed.io = server ? io.attach(server) : io
1950   return composed
1951 }
1952
1953 /**
1954  * NOME DO ARQUIVO
1955  * ./services/socketio/index.test.js
1956  */
1957 import http from 'http'
1958 import ioclient from 'socket.io-client'
1959
1960 import socketio from './'
1961
1962 let server
1963 let addr
1964 let io
1965
1966 beforeAll(done => {
1967   server = http.createServer()
1968   addr = server.listen().address()
1969   io = socketio()({ server }).io
1970   done()
1971 })
1972
1973 afterAll(done => {
1974   io.close()
1975   server.close()
1976   done()
1977 })
1978
1979 describe('[service] socket.io', () => {
1980   test('connects client to the server', done => {
1981     const mock = jest.fn()
1982     io.on('connection', mock)
```

```
1983      const socketclient = ioclient.connect(`http://[${addr.address
              }]:${addr.port}`)
1984
1985      socketclient
1986        .on('connect', () => {
1987          expect(mock).toHaveBeenCalled()
1988          socketclient.disconnect()
1989          done()
1990        })
1991    })
1992
1993    describe('communicate', () => {
1994      let socketclient
1995
1996      beforeEach(done => {
1997        socketclient = ioclient.connect(`http://[${addr.address}]:$
              {addr.port}`)
1998        socketclient.on('connect', () => {
1999          done()
2000        })
2001      })
2002
2003      afterEach((done) => {
2004        if (socketclient.connected) {
2005          socketclient.disconnect()
2006        }
2007        done()
2008      })
2009
2010      test('sends ACK', done => {
2011        io.emit('handshake', 'ACK')
2012
2013        socketclient.on('handshake', (message, answer) => {
2014          expect(message).toEqual('ACK')
2015          done()
2016        })
2017      })
2018    })
2019  })
2020
2021  /**
2022   * NOME DO ARQUIVO
2023   * ./services/db/migrations/20181018232751_create_service_calls.
          js
2024   */
2025  exports.up = function (knex, Promise) {
```

```
2026   return knex.schema.createTable('service_calls', function (table
           ) {
2027     table.uuid('id').primary()
2028     table.string('description')
2029     table.integer('call_rating').defaultTo(0)
2030     table.integer('service_rating').defaultTo(0)
2031     table.boolean('is_solved')
2032     table.timestamp('started_at')
2033     table.timestamp('ended_at')
2034     table.integer('duration').comment('Service duration in
             milliseconds')
2035     table.timestamps()
2036
2037     table.uuid('customer_id')
2038       .references('customers.id')
2039       .onDelete('SET NULL')
2040     table.uuid('user_id')
2041       .references('users.id')
2042       .onDelete('SET NULL')
2043     table.index('user_id')
2044   })
2045 }
2046
2047 exports.down = function (knex, Promise) {
2048   return knex.schema.dropTable('service_calls')
2049 }
2050
2051 /**
2052  * NOME DO ARQUIVO
2053  * ./services/db/migrations/20181016214442_create_customers.js
2054  */
2055 exports.up = function (knex, Promise) {
2056   return knex.schema.createTable('customers', function (table) {
2057     table.uuid('id').primary()
2058     table.string('email').notNullable().unique()
2059     table.string('name')
2060     table.timestamps()
2061   })
2062 }
2063
2064 exports.down = function (knex, Promise) {
2065   return knex.schema.dropTable('customers')
2066 }
2067
2068 /**
2069  * NOME DO ARQUIVO
```

```
2070  * ./services/db/migrations/20181019000236
          _alter_customer_add_email_index.js
2071  */
2072 exports.up = function (knex, Promise) {
2073   return knex.schema.table('customers', function (table) {
2074     table.index('email')
2075   })
2076 }
2077
2078 exports.down = function (knex, Promise) {
2079   return knex.schema.table('customers', function (table) {
2080     table.dropIndex('email')
2081   })
2082 }
2083
2084 /**
2085  * NOME DO ARQUIVO
2086  * ./services/db/migrations/20181019212913
          _create_datediff_functions.js
2087  */
2088 exports.up = function (knex, Promise) {
2089   return knex.raw(`
2090     CREATE OR REPLACE FUNCTION date_diff(units VARCHAR(30),
           start_t TIMESTAMP WITH TIME ZONE, end_t TIMESTAMP WITH
           TIME ZONE)
2091       RETURNS INT AS $$
2092     DECLARE
2093       diff_interval INTERVAL;
2094       diff INT = 0;
2095     BEGIN
2096       -- Minus operator returns interval 'DDD days HH:MI:SS'
2097       diff_interval = end_t - start_t;
2098
2099       diff = DATE_PART('hour', diff_interval);
2100
2101       IF units IN ('hh', 'hour') THEN
2102         RETURN diff;
2103       END IF;
2104
2105       diff = diff * 60 + DATE_PART('minute', diff_interval);
2106
2107       IF units IN ('mi', 'n', 'minute') THEN
2108         RETURN diff;
2109       END IF;
2110
2111       diff = diff * 60 + DATE_PART('second', diff_interval);
2112
```

```
2113        IF units IN ('se', 's', 'second') THEN
2114          RETURN diff;
2115        END IF;
2116
2117        diff = diff * 1000 + DATE_PART('milliseconds',
                diff_interval);
2118
2119        RETURN diff;
2120      END;
2121      $$ LANGUAGE plpgsql;
2122    `)
2123  }
2124
2125  exports.down = function (knex, Promise) {
2126    return knex.raw('DROP FUNCTION IF EXISTS date_diff;')
2127  }
2128
2129  /**
2130   * NOME DO ARQUIVO
2131   * ./services/db/migrations/20180701202709_create_users.js
2132   */
2133  exports.up = function (knex, Promise) {
2134    return knex.schema.createTable('users', function (table) {
2135      table.uuid('id').primary()
2136      table.string('email').notNullable().unique()
2137      table.string('password').notNullable()
2138      table.string('name')
2139      table.enum('role', ['admin', 'user']).defaultTo('user')
2140      table.timestamps()
2141    })
2142  }
2143
2144  exports.down = function (knex, Promise) {
2145    return knex.schema.dropTable('users')
2146  }
2147
2148  /**
2149   * NOME DO ARQUIVO
2150   * ./services/db/migrations/20181019233435
          _bind_set_duration_trigger_to_service_call.js
2151   */
2152  exports.up = function (knex, Promise) {
2153    return knex.raw(`
2154      CREATE TRIGGER set_duration_service_call
2155      BEFORE INSERT OR UPDATE OF ended_at
2156      ON service_calls
2157      FOR EACH ROW
```

```
2158      EXECUTE PROCEDURE set_duration();
2159    `)
2160  }
2161
2162  exports.down = function (knex, Promise) {
2163    return knex.raw('DROP TRIGGER IF EXISTS
          set_duration_service_call ON service_calls; ')
2164  }
2165
2166  /**
2167   * NOME DO ARQUIVO
2168   * ./services/db/migrations/20181019231013
          _create_set_duration_trigger.js
2169   */
2170  exports.up = function (knex, Promise) {
2171    return knex.raw(`
2172      CREATE OR REPLACE FUNCTION set_duration()
2173        RETURNS TRIGGER AS $$
2174      BEGIN
2175        IF NEW.ended_at IS NOT NULL THEN
2176          NEW.duration = date_diff('ms', NEW.started_at, NEW.
                ended_at);
2177        END IF;
2178        RETURN NEW;
2179      END;
2180      $$ LANGUAGE plpgsql;
2181    `)
2182  }
2183
2184  exports.down = function (knex, Promise) {
2185    return knex.raw('DROP FUNCTION IF EXISTS set_duration;')
2186  }
2187
2188  /**
2189   * NOME DO ARQUIVO
2190   * ./services/db/index.js
2191   */
2192  import Knex from 'knex'
2193  import { knexSnakeCaseMappers, Model } from 'objection'
2194  import { knex as knexConfig } from '../../config'
2195
2196  let knex = null
2197
2198  export function init () {
2199    if (knex) {
2200      throw new Error("There's already a database connection.")
2201    }
```

```
2202   knex = Knex({
2203     ...knexConfig,
2204     ...knexSnakeCaseMappers()
2205   })
2206   Model.knex(knex)
2207   return knex
2208 }
2209
2210 export async function destroy () {
2211   if (!knex) {
2212     throw new Error("There's no database connection.")
2213   }
2214   await knex.destroy()
2215   const conn = knex
2216   knex = null
2217   return conn
2218 }
2219
2220 export function get () {
2221   if (!knex) {
2222     throw new Error("There's no database connection.")
2223   }
2224   return knex
2225 }
2226
2227 export default {
2228   init,
2229   destroy,
2230   get
2231 }
2232
2233 /**
2234  * NOME DO ARQUIVO
2235  * ./services/db/seeds/01_create_users.js
2236  */
2237 require('babel-core/register') // necessary to execute all
       imports with ES6 syntax
2238
2239 const Model = require('objection').Model
2240 const createUsersDry = require('../fixtures/create-user').
       createUsersDry
2241
2242 const ID = {
2243   admin: '656f3c98-0051-491c-80f4-03b9aef47042',
2244   userOne: '961e8a22-3399-442f-8430-e9e0bb2aad64',
2245   userTwo: '1b2d29fa-1205-4199-b7bc-cda352479ac8'
2246 }
```

```
2247
2248 const users = createUsersDry([
2249   {
2250     id: ID.admin,
2251     email: 'admin@sac.com',
2252     password: '123456',
2253     role: 'admin'
2254   },
2255   {
2256     id: ID.userOne,
2257     email: 'user1@sac.com',
2258     password: '123456'
2259   },
2260   {
2261     id: ID.userTwo,
2262     email: 'user2@sac.com',
2263     password: '123456'
2264   }
2265 ])
2266
2267 exports.ID = ID
2268 exports.seed = function (knex, Promise) {
2269   Model.knex(knex)
2270   const User = require('../../../api/users/model').default
2271   return User
2272     .query()
2273     .delete()
2274     .then(() => {
2275       return User.query().insert(users)
2276     })
2277 }
2278
2279 /**
2280  * NOME DO ARQUIVO
2281  * ./services/db/seeds/02_create_customers.js
2282  */
2283 require('babel-core/register') // necessary to execute all
       imports with ES6 syntax
2284
2285 const Model = require('objection').Model
2286 const createCustomersDry = require('../fixtures/create-customer')
       .createCustomersDry
2287
2288 const ID = {
2289   customerOne: '0003736f-3ac6-4695-bf3e-309bf1d895ee',
2290   customerTwo: '51daa685-5016-4f75-ba0b-1b07f1b3009f',
2291   customerThree: '5675f188-53c7-4321-adbd-3f604077120d'
```

```
2292 }
2293
2294 const customers = createCustomersDry([
2295   {
2296     id: ID.customerOne,
2297     email: 'customer1@sac.com'
2298   },
2299   {
2300     id: ID.customerTwo,
2301     email: 'customer2@sac.com'
2302   },
2303   {
2304     id: ID.customerThree,
2305     email: 'customer3@sac.com'
2306   }
2307 ])
2308
2309 exports.ID = ID
2310 exports.seed = function (knex, Promise) {
2311   Model.knex(knex)
2312   const Customer = require('../../../api/customers/model').
          default
2313   return Customer
2314     .query()
2315     .delete()
2316     .then(() => {
2317       return Customer.query().insert(customers)
2318     })
2319 }
2320
2321 /**
2322  * NOME DO ARQUIVO
2323  * ./services/db/seeds/03_create_service_calls.js
2324  */
2325 require('babel-core/register') // necessary to execute all
       imports with ES6 syntax
2326
2327 const Model = require('objection').Model
2328 const createServiceCallsDry = require('../fixtures/create-service
       -call')
2329   .createServiceCallsDry
2330
2331 const USER_ID = require('./01_create_users').ID
2332 const CUSTOMER_ID = require('./02_create_customers').ID
2333
2334 const adminServiceCalls = createServiceCallsDry([
2335   {
```

```
2336      userId: USER_ID.admin,
2337      customerId: CUSTOMER_ID.customerOne
2338    },
2339    {
2340      userId: USER_ID.admin,
2341      customerId: CUSTOMER_ID.customerTwo
2342    },
2343    {
2344      userId: USER_ID.admin,
2345      customerId: CUSTOMER_ID.customerThree
2346    },
2347    {
2348      userId: USER_ID.admin,
2349      customerId: CUSTOMER_ID.customerOne
2350    },
2351    {
2352      userId: USER_ID.admin,
2353      customerId: CUSTOMER_ID.customerTwo
2354    },
2355    {
2356      userId: USER_ID.admin,
2357      customerId: CUSTOMER_ID.customerThree
2358    }
2359  ])
2360  const userOneServicesCalls = createServiceCallsDry([
2361    {
2362      userId: USER_ID.userOne,
2363      customerId: CUSTOMER_ID.customerOne
2364    },
2365    {
2366      userId: USER_ID.userOne,
2367      customerId: CUSTOMER_ID.customerTwo
2368    },
2369    {
2370      userId: USER_ID.userOne,
2371      customerId: CUSTOMER_ID.customerThree
2372    }
2373  ])
2374  // const serviceCalls = createServiceCallsDry([
2375  //   {
2376  //     customerId: CUSTOMER_ID.customerThree
2377  //   }
2378  // ])
2379
2380  exports.seed = function (knex, Promise) {
2381    Model.knex(knex)
```

```
2382   const ServiceCall = require('../../../api/service-calls/model')
           .default
2383   return ServiceCall.query()
2384     .delete()
2385     .then(() => {
2386       return ServiceCall.query().insert([
2387         ...adminServiceCalls,
2388         ...userOneServicesCalls
2389       ])
2390     })
2391 }
2392
2393 /**
2394  * NOME DO ARQUIVO
2395  * ./services/db/fixtures/create-customer.js
2396  */
2397 import faker from 'faker'
2398 import Customer from '../../../api/customers/model'
2399
2400 export const createCustomerDry = ({
2401   id = faker.random.uuid(),
2402   email = faker.internet.email(),
2403   name = faker.name.findName()
2404 } = {}) => ({
2405   id, email, name
2406 })
2407
2408 export const createCustomersDry = data =>
2409   data.map(customer => createCustomerDry(customer))
2410
2411 export const createCustomer = data =>
2412   Customer.query().insert(createCustomerDry(data)).returning('*')
2413
2414 export const createCustomers = data =>
2415   Customer.query().insert(createCustomersDry(data)).returning('*'
           )
2416
2417 /**
2418  * NOME DO ARQUIVO
2419  * ./services/db/fixtures/create-user.js
2420  */
2421 import faker from 'faker'
2422 import User from '../../../api/users/model'
2423
2424 export const createUserDry = ({
2425   id = faker.random.uuid(),
2426   email = faker.internet.email(),
```

```
2427   name = faker.name.findName(),
2428   password = '123456',
2429   role = 'user'
2430 } = {}) => ({
2431   id,
2432   email,
2433   name,
2434   password,
2435   role
2436 })
2437
2438 export const createUsersDry = data => data.map(user =>
       createUserDry(user))
2439
2440 export const createUser = data =>
2441   User.query().insert(createUserDry(data)).returning('*')
2442
2443 export const createUsers = data =>
2444   User.query().insert(createUsersDry(data)).returning('*')
2445
2446 /**
2447  * NOME DO ARQUIVO
2448  * ./services/db/fixtures/index.js
2449  */
2450 export { createCustomer, createCustomers } from './create-
       customer'
2451 export { createServiceCall, createServiceCalls } from './create-
       service-call'
2452 export { createUser, createUsers } from './create-user'
2453
2454 /**
2455  * NOME DO ARQUIVO
2456  * ./services/db/fixtures/create-service-call.js
2457  */
2458 import faker from 'faker'
2459 import addMinutes from 'date-fns/fp/addMinutes'
2460 import ServiceCall from '../../../api/service-calls/model'
2461
2462 const startedAtDate = new Date()
2463
2464 export const createServiceCallDry = ({
2465   id = faker.random.uuid(),
2466   description = faker.lorem.sentences(3),
2467   callRating = faker.random.number(4) + 1,
2468   serviceRating = faker.random.number(4) + 1,
2469   isSolved = faker.random.boolean(),
2470   startedAt = startedAtDate,
```

```
2471   endedAt = addMinutes(faker.random.number(120), startedAtDate),
2472   customerId,
2473   userId
2474 } = {}) => ({
2475   id,
2476   description,
2477   ...(faker.random.number(3) > 1 ? {
2478     callRating: (userId && endedAt) ? callRating : 0,
2479     serviceRating: (userId && endedAt) ? serviceRating : 0,
2480     isSolved: (userId && endedAt) ? isSolved : undefined
2481   } : {
2482     callRating: 0,
2483     serviceRating: 0,
2484     isSolved: undefined
2485   }),
2486   startedAt: userId ? startedAt : undefined,
2487   endedAt: userId ? endedAt : undefined,
2488   customerId,
2489   userId
2490 })
2491
2492 export const createServiceCallsDry = data =>
2493   data.map(serviceCall => createServiceCallDry(serviceCall))
2494
2495 export const createServiceCall = data =>
2496   ServiceCall.query().insert(createServiceCallDry(data)).
       returning('*')
2497
2498 export const createServiceCalls = data =>
2499   ServiceCall.query().insert(createServiceCallsDry(data)).
       returning('*')
```

Listing 7.1 – Projeto SAC Api

7.2 SAC MANAGER

```
1
2 /**
3  * NOME DO ARQUIVO
4  * ./constants.js
5  */
6 export const AUTH_TOKEN_KEY = 'jwt_token'; // eslint-disable-line
```

```
7
8  /**
9   * NOME DO ARQUIVO
10  * ./setupProxy.js
11  */
12 const proxy = require('http-proxy-middleware'); // eslint-disable
       -line
13
14 module.exports = function setupProxy(app) {
15   app.use(proxy('/api', { target: 'http://localhost:6000/' }));
16   app.use(proxy('/events', { target: 'ws://localhost:6000/', ws:
         true }));
17 };
18
19 /**
20  * NOME DO ARQUIVO
21  * ./index.js
22  */
23 import React from 'react';
24 import { render } from 'react-dom';
25
26 import Root from './Root';
27
28 render(<Root />, document.getElementById('root'));
29
30 /**
31  * NOME DO ARQUIVO
32  * ./utils/socket-middleware.js
33  */
34 import { types } from 'src/actions/socket';
35
36 const {
37   SOCKET_OPEN_REQUEST,
38   SOCKET_OPEN_SUCCESS,
39   SOCKET_CLOSE_REQUEST,
40   SOCKET_CLOSE_SUCCESS,
41   SOCKET_CONN_ERROR,
42   FSA,
43   RSSA,
44 } = types;
45
46 const createSocketMiddleware = socket => ({ dispatch }) => {
47   socket.on(FSA, dispatch);
48
49   socket.on('connect', () =>
50     dispatch({
51       type: SOCKET_OPEN_SUCCESS,
```

```
52        payload: { sid: socket.id },
53      })
54    );
55    socket.on('disconnect', () => dispatch({ type:
          SOCKET_CLOSE_SUCCESS }));
56    socket.on('error', error =>
57      dispatch({ type: SOCKET_CONN_ERROR, payload: { error } })
58    );
59
60    return next => action => {
61      const socketAction = action[RSSA];
62
63      if (socketAction) {
64        const { ack, event, message, optimistic } = socketAction;
65        message.meta = {
66          ...message.meta,
67          sid: socket.id,
68        };
69        socket.emit(event || FSA, message, ack);
70        if (optimistic) {
71          dispatch(message);
72        }
73        return;
74      }
75
76      switch (action.type) {
77        case SOCKET_OPEN_REQUEST:
78          socket.connect();
79          break;
80        case SOCKET_CLOSE_REQUEST:
81          socket.disconnect();
82          break;
83        case SOCKET_CONN_ERROR:
84          socket.connect();
85          break;
86        default:
87          break;
88      }
89
90      next(action);
91    };
92  };
93
94  export default createSocketMiddleware;
95
96  /**
97   * NOME DO ARQUIVO
```

```
98  * ./utils/auth-api-middleware.js
99  */
100 import { RSAA } from 'redux-api-middleware';
101
102 import { AUTH_TOKEN_KEY } from 'src/constants';
103 import els from 'src/utils/expirable-local-storage';
104
105 const apiRoot = process.env.REACT_APP_API_ROOT || '';
106
107 const authApiMiddleware = () => next => action => {
108   const apiAction = action[RSAA];
109
110   if (apiAction) {
111     const jwtToken = els.get(AUTH_TOKEN_KEY);
112     if (jwtToken) {
113       apiAction.headers = {
114         Authorization: `Bearer ${jwtToken}`,
115       };
116     }
117     apiAction.endpoint = apiRoot.concat(apiAction.endpoint);
118   }
119
120   next(action);
121 };
122
123 export default authApiMiddleware;
124
125 /**
126  * NOME DO ARQUIVO
127  * ./utils/expirable-local-storage.js
128  */
129 import ls from 'local-storage';
130
131 export default {
132   get: key => {
133     const record = JSON.parse(ls.get(key));
134     if (!record) {
135       return false;
136     }
137     return new Date().getTime() < record.timestamp && record.
          value;
138   },
139   set: (key, value, min = 120) => {
140     const ms = min * 60 * 1000;
141     const record = {
142       value,
143       timestamp: new Date().getTime() + ms,
```

```
144       };
145       return ls.set(key, JSON.stringify(record));
146     },
147   };
148
149   /**
150    * NOME DO ARQUIVO
151    * ./utils/immer-reducer.js
152    */
153   import produce from 'immer';
154
155   const immerReducer = (actionsMap, defaultState) => (
156     state = defaultState,
157     { type, payload }
158   ) =>
159     produce(state, draft => {
160       const action = actionsMap[type];
161       action && action(draft, payload); // eslint-disable-line
162     });
163
164   export default immerReducer;
165
166   /**
167    * NOME DO ARQUIVO
168    * ./utils/rtc-middleware.js
169    */
170   import Peer from 'simple-peer';
171
172   import {
173     disconnectPeer,
174     setPeer,
175     setStream,
176     sendSignal,
177     types,
178   } from 'src/actions/rtc';
179
180   const { PEER_CONNECT, PEER_DISCONNECT, SIGNAL_RECEIVE } = types;
181
182   const rtcMiddleware = ({ dispatch }) => {
183     let peer;
184
185     return next => action => {
186       const { meta, payload, type } = action;
187       switch (type) {
188         case PEER_CONNECT: {
189           const { stream } = payload;
190           const { room } = meta;
```

```
191
192         peer = new Peer({ initiator: true, stream, trickle: false
                });
193
194         peer.on('signal', signal => dispatch(sendSignal({ room,
                signal })));
195         peer.on('connect', () => dispatch(setPeer(peer)));
196         peer.on('stream', peerStream => dispatch(setStream(
                peerStream)));
197
198         peer.on('close', () => next(disconnectPeer()));
199         peer.on('error', err => dispatch(disconnectPeer({ err }))
                );
200
201         break;
202       }
203       case PEER_DISCONNECT: {
204         peer.destroy();
205         return;
206       }
207       case SIGNAL_RECEIVE: {
208         const { signal } = payload;
209         peer.signal(signal);
210         break;
211       }
212       default:
213         break;
214     }
215
216     next(action);
217   };
218 };
219
220 export default rtcMiddleware;
221
222 /**
223  * NOME DO ARQUIVO
224  * ./utils/create-constants.js
225  */
226 export default (prefix = '') => (...keys) =>
227   keys.reduce((obj, key) => ({ ...obj, [key]: prefix + key }),
        {});
228
229 /**
230  * NOME DO ARQUIVO
231  * ./styles/InlineFlex.js
232  */
```

```
233 import styled from 'styled-components';
234 import { Flex } from '@rebass/grid';
235 import { textAlign } from 'styled-system';
236
237 const InlineFlex = styled(Flex)`
238   ${textAlign};
239   display: inline-flex;
240 `;
241
242 export default InlineFlex;
243
244 /**
245  * NOME DO ARQUIVO
246  * ./styles/Global.js
247  */
248 /* stylelint-disable */
249 import { createGlobalStyle } from 'styled-components';
250
251 const Global = createGlobalStyle`
252   /*! normalize.css v8.0.0 | MIT License | github.com/necolas/
        normalize.css */
253
254   /* Document
255     ========================================================================
          */
256
257   /**
258   * 1. Correct the line height in all browsers.
259   * 2. Prevent adjustments of font size after orientation changes
        in iOS.
260   */
261
262   html {
263     line-height: 1.15; /* 1 */
264     -webkit-text-size-adjust: 100%; /* 2 */
265   }
266
267   /* Sections
268     ========================================================================
          */
269
270   /**
271   * Remove the margin in all browsers.
272   */
273
274   body {
275     margin: 0;
```

```
276   }
277
278   /**
279   * Correct the font size and margin on h1 elements within
          section and
280   * article contexts in Chrome, Firefox, and Safari.
281   */
282
283   h1 {
284     font-size: 2em;
285     margin: 0.67em 0;
286   }
287
288   /* Grouping content
289     ========================================================================
          */
290
291   /**
292   * 1. Add the correct box sizing in Firefox.
293   * 2. Show the overflow in Edge and IE.
294   */
295
296   hr {
297     box-sizing: content-box; /* 1 */
298     height: 0; /* 1 */
299     overflow: visible; /* 2 */
300   }
301
302   /**
303   * 1. Correct the inheritance and scaling of font size in all
          browsers.
304   * 2. Correct the odd em font sizing in all browsers.
305   */
306
307   pre {
308     font-family: monospace, monospace; /* 1 */
309     font-size: 1em; /* 2 */
310   }
311
312   /* Text-level semantics
313     ========================================================================
          */
314
315   /**
316   * Remove the gray background on active links in IE 10.
317   */
318
```

```
319  a {
320    background-color: transparent;
321  }
322
323  /**
324  * 1. Remove the bottom border in Chrome 57-
325  * 2. Add the correct text decoration in Chrome, Edge, IE, Opera
       , and Safari.
326  */
327
328  abbr[title] {
329    border-bottom: none; /* 1 */
330    text-decoration: underline; /* 2 */
331    text-decoration: underline dotted; /* 2 */
332  }
333
334  /**
335  * Add the correct font weight in Chrome, Edge, and Safari.
336  */
337
338  b,
339  strong {
340    font-weight: bolder;
341  }
342
343  /**
344  * 1. Correct the inheritance and scaling of font size in all
       browsers.
345  * 2. Correct the odd em font sizing in all browsers.
346  */
347
348  code,
349  kbd,
350  samp {
351    font-family: monospace, monospace; /* 1 */
352    font-size: 1em; /* 2 */
353  }
354
355  /**
356  * Add the correct font size in all browsers.
357  */
358
359  small {
360    font-size: 80%;
361  }
362
363  /**
```

```
364   * Prevent sub and sup elements from affecting the line height
          in
365   * all browsers.
366   */
367
368   sub,
369   sup {
370     font-size: 75%;
371     line-height: 0;
372     position: relative;
373     vertical-align: baseline;
374   }
375
376   sub {
377     bottom: -0.25em;
378   }
379
380   sup {
381     top: -0.5em;
382   }
383
384   /* Embedded content
385     ========================================================================
          */
386
387   /**
388   * Remove the border on images inside links in IE 10.
389   */
390
391   img {
392     border-style: none;
393   }
394
395   /* Forms
396     ========================================================================
          */
397
398   /**
399   * 1. Change the font styles in all browsers.
400   * 2. Remove the margin in Firefox and Safari.
401   */
402
403   button,
404   input,
405   optgroup,
406   select,
407   textarea {
```

```
408     font-family: inherit; /* 1 */
409     font-size: 100%; /* 1 */
410     line-height: 1.15; /* 1 */
411     margin: 0; /* 2 */
412   }
413
414   /**
415   * Show the overflow in IE.
416   * 1. Show the overflow in Edge.
417   */
418
419   button,
420   input { /* 1 */
421     overflow: visible;
422   }
423
424   /**
425   * Remove the inheritance of text transform in Edge, Firefox,
        and IE.
426   * 1. Remove the inheritance of text transform in Firefox.
427   */
428
429   button,
430   select { /* 1 */
431     text-transform: none;
432   }
433
434   /**
435   * Correct the inability to style clickable types in iOS and
        Safari.
436   */
437
438   button,
439   [type="button"],
440   [type="reset"],
441   [type="submit"] {
442     -webkit-appearance: button;
443   }
444
445   /**
446   * Remove the inner border and padding in Firefox.
447   */
448
449   button::-moz-focus-inner,
450   [type="button"]::-moz-focus-inner,
451   [type="reset"]::-moz-focus-inner,
452   [type="submit"]::-moz-focus-inner {
```

```
453     border-style: none;
454     padding: 0;
455   }
456
457   /**
458   * Restore the focus styles unset by the previous rule.
459   */
460
461   button:-moz-focusring,
462   [type="button"]:-moz-focusring,
463   [type="reset"]:-moz-focusring,
464   [type="submit"]:-moz-focusring {
465     outline: 1px dotted ButtonText;
466   }
467
468   /**
469   * Correct the padding in Firefox.
470   */
471
472   fieldset {
473     padding: 0.35em 0.75em 0.625em;
474   }
475
476   /**
477   * 1. Correct the text wrapping in Edge and IE.
478   * 2. Correct the color inheritance from fieldset elements in IE
        .
479   * 3. Remove the padding so developers are not caught out when
        they zero out
480   *    fieldset elements in all browsers.
481   */
482
483   legend {
484     box-sizing: border-box; /* 1 */
485     color: inherit; /* 2 */
486     display: table; /* 1 */
487     max-width: 100%; /* 1 */
488     padding: 0; /* 3 */
489     white-space: normal; /* 1 */
490   }
491
492   /**
493   * Add the correct vertical alignment in Chrome, Firefox, and
        Opera.
494   */
495
496   progress {
```

```
497     vertical-align: baseline;
498   }
499
500   /**
501   * Remove the default vertical scrollbar in IE 10+.
502   */
503
504   textarea {
505     overflow: auto;
506   }
507
508   /**
509   * 1. Add the correct box sizing in IE 10.
510   * 2. Remove the padding in IE 10.
511   */
512
513   [type="checkbox"],
514   [type="radio"] {
515     box-sizing: border-box; /* 1 */
516     padding: 0; /* 2 */
517   }
518
519   /**
520   * Correct the cursor style of increment and decrement buttons
        in Chrome.
521   */
522
523   [type="number"]::-webkit-inner-spin-button,
524   [type="number"]::-webkit-outer-spin-button {
525     height: auto;
526   }
527
528   /**
529   * 1. Correct the odd appearance in Chrome and Safari.
530   * 2. Correct the outline style in Safari.
531   */
532
533   [type="search"] {
534     -webkit-appearance: textfield; /* 1 */
535     outline-offset: -2px; /* 2 */
536   }
537
538   /**
539   * Remove the inner padding in Chrome and Safari on macOS.
540   */
541
542   [type="search"]::-webkit-search-decoration {
```

```
543     -webkit-appearance: none;
544   }
545
546   /**
547   * 1. Correct the inability to style clickable types in iOS and
          Safari.
548   * 2. Change font properties to inherit in Safari.
549   */
550
551   ::-webkit-file-upload-button {
552     -webkit-appearance: button; /* 1 */
553     font: inherit; /* 2 */
554   }
555
556   /* Interactive
557     ========================================================================
          */
558
559   /*
560   * Add the correct display in Edge, IE 10+, and Firefox.
561   */
562
563   details {
564     display: block;
565   }
566
567   /*
568   * Add the correct display in all browsers.
569   */
570
571   summary {
572     display: list-item;
573   }
574
575   /* Misc
576     ========================================================================
          */
577
578   /**
579   * Add the correct display in IE 10+.
580   */
581
582   template {
583     display: none;
584   }
585
586   /**
```

```
587   * Add the correct display in IE 10.
588   */
589
590   [hidden] {
591     display: none;
592   }
593
594   /********************************
595               Page
596   ********************************/
597
598   html,
599   body {
600     height: 100%;
601     width: 100%;
602   }
603
604   html {
605     font-size: 14px;
606   }
607
608   body {
609     margin: 0px;
610     padding: 0px;
611     overflow-x: hidden;
612     min-width: 320px;
613     background: #FFFFFF;
614     font-family: 'Lato', 'Helvetica Neue', Arial, Helvetica, sans
          -serif;
615     font-size: 14px;
616     line-height: 1.4285em;
617     color: rgba(0, 0, 0, 0.87);
618     font-smoothing: antialiased;
619   }
620
621   /********************************
622               Headers
623   ********************************/
624
625   h1,
626   h2,
627   h3,
628   h4,
629   h5 {
630     font-family: 'Lato', 'Helvetica Neue', Arial, Helvetica, sans
          -serif;
631     line-height: 1.28571429em;
```

```
632     margin: calc(2rem -  0.14285714em ) 0em 1rem;
633     font-weight: bold;
634     padding: 0em;
635   }
636
637   h1 {
638     min-height: 1rem;
639     font-size: 2rem;
640   }
641
642   h2 {
643     font-size: 1.71428571rem;
644   }
645
646   h3 {
647     font-size: 1.28571429rem;
648   }
649
650   h4 {
651     font-size: 1.07142857rem;
652   }
653
654   h5 {
655     font-size: 1rem;
656   }
657
658   h1:first-child,
659   h2:first-child,
660   h3:first-child,
661   h4:first-child,
662   h5:first-child {
663     margin-top: 0em;
664   }
665
666   h1:last-child,
667   h2:last-child,
668   h3:last-child,
669   h4:last-child,
670   h5:last-child {
671     margin-bottom: 0em;
672   }
673
674   /*******************************
675                Text
676   *******************************/
677
678   p {
```

```
679     margin: 0em 0em 1em;
680     line-height: 1.4285em;
681   }
682
683   p:first-child {
684     margin-top: 0em;
685   }
686
687   p:last-child {
688     margin-bottom: 0em;
689   }
690
691   /*-------------------
692           Links
693   --------------------*/
694
695   a {
696     color: #4183C4;
697     text-decoration: none;
698   }
699
700   a:hover {
701     color: #1e70bf;
702     text-decoration: none;
703   }
704
705   /*******************************
706           Scrollbars
707   *******************************/
708
709   /*******************************
710             Highlighting
711   *******************************/
712
713   /* Site */
714
715   ::-webkit-selection {
716     background-color: #CCE2FF;
717     color: rgba(0, 0, 0, 0.87);
718   }
719
720   ::-moz-selection {
721     background-color: #CCE2FF;
722     color: rgba(0, 0, 0, 0.87);
723   }
724
725   ::selection {
```

```
726     background-color: #CCE2FF;
727     color: rgba(0, 0, 0, 0.87);
728   }
729
730   /* Form */
731
732   textarea::-webkit-selection,
733   input::-webkit-selection {
734     background-color: rgba(100, 100, 100, 0.4);
735     color: rgba(0, 0, 0, 0.87);
736   }
737
738   textarea::-moz-selection,
739   input::-moz-selection {
740     background-color: rgba(100, 100, 100, 0.4);
741     color: rgba(0, 0, 0, 0.87);
742   }
743
744   textarea::selection,
745   input::selection {
746     background-color: rgba(100, 100, 100, 0.4);
747     color: rgba(0, 0, 0, 0.87);
748   }
749
750   /* Force Simple Scrollbars */
751
752   body ::-webkit-scrollbar {
753     -webkit-appearance: none;
754     width: 10px;
755     height: 10px;
756   }
757
758   body ::-webkit-scrollbar-track {
759     background: rgba(0, 0, 0, 0.1);
760     border-radius: 0px;
761   }
762
763   body ::-webkit-scrollbar-thumb {
764     cursor: pointer;
765     border-radius: 5px;
766     background: rgba(0, 0, 0, 0.25);
767     -webkit-transition: color 0.2s ease;
768     transition: color 0.2s ease;
769   }
770
771   body ::-webkit-scrollbar-thumb:window-inactive {
772     background: rgba(0, 0, 0, 0.15);
```

```
773   }
774
775   body ::-webkit-scrollbar-thumb:hover {
776     background: rgba(128, 135, 139, 0.8);
777   }
778
779   #root, #app {
780     height: 100%;
781     width: 100%;
782   }
783 `;
784
785 export default Global;
786
787 /**
788  * NOME DO ARQUIVO
789  * ./styles/NavLinkBox.js
790  */
791 import styled from 'styled-components';
792 import { NavLink } from 'react-router-dom';
793
794 const NavLinkBox = styled(NavLink)`
795   color: #ffffff;
796
797   .service-call__rating {
798     display: none;
799   }
800
801   .service-call__solved {
802     display: flex;
803   }
804
805   &.active {
806     background-color: rgba(221, 221, 221, 0.4);
807     color: #ffffff;
808   }
809
810   &:hover {
811     background-color: rgba(221, 221, 221, 0.4);
812     color: #ffffff;
813
814     .service-call__rating {
815       display: flex;
816     }
817
818     .service-call__solved {
819       display: none;
```

```
820     }
821   }
822 `;
823
824 export default NavLinkBox;
825
826 /**
827  * NOME DO ARQUIVO
828  * ./styles/__tests__/Button.js
829  */
830 import React from 'react';
831 import { render } from 'react-testing-library';
832
833 import Button from '../Button';
834
835 describe('[styled-component] Button', () => {
836   it('matches snapshot', () => {
837     const { container } = render(<Button />);
838     expect(container.firstChild).toMatchSnapshot();
839   });
840 });
841
842 /**
843  * NOME DO ARQUIVO
844  * ./styles/Button.js
845  */
846 import styled from 'styled-components';
847
848 const Button = styled.button`
849   font-size: 1em;
850   margin: 1em;
851   padding: 0.25em 1em;
852   border-radius: 3px;
853   color: ${props => props.theme.main};
854   border: 2px solid ${props => props.theme.main};
855 `;
856
857 export default Button;
858
859 /**
860  * NOME DO ARQUIVO
861  * ./components/LoginForm/index.js
862  */
863 import LoginForm from './components/LoginForm';
864
865 export default LoginForm;
866
```

```
867 /**
868  * NOME DO ARQUIVO
869  * ./components/LoginForm/utils/validation.js
870  */
871 export const validationProps = (fieldName, errors, touched) => {
872   const isTouched = touched[fieldName];
873   const fieldErrors = errors[fieldName];
874
875   if (fieldErrors && isTouched) {
876     return {
877       hasFeedback: true,
878       help: fieldErrors,
879       validateStatus: 'error',
880     };
881   }
882
883   if (isTouched) {
884     return {
885       hasFeedback: true,
886       validateStatus: 'success',
887     };
888   }
889
890   return null;
891 };
892
893 export const validateForm = values => {
894   const errors = {};
895
896   if (!values.email) {
897     errors.email = 'Email is required';
898   } else if (!/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i.test(
       values.email)) {
899     errors.email = 'Invalid email address';
900   }
901
902   if (!values.password) {
903     errors.password = 'Password is required';
904   } else if (values.password.length < 6) {
905     errors.password = 'Minimum password size is 6';
906   }
907
908   return errors;
909 };
910
911 /**
912  * NOME DO ARQUIVO
```

```
913  * ./components/LoginForm/components/LoginForm.js
914  */
915 import React from 'react';
916 import { Formik } from 'formik';
917 import PropTypes from 'prop-types';
918 import { compose, withHandlers } from 'recompose';
919 import { connect } from 'react-redux';
920 import { Button, Box, FormField, TextInput, Text } from 'grommet'
        ;
921
922 import * as actions from 'src/actions/auth';
923 import { validateForm } from '../utils/validation';
924
925 const LoginForm = ({ authMessage, onSubmit }) => (
926   <Formik
927     initialValues={{ email: '', password: '' }}
928     validate={validateForm}
929     onSubmit={onSubmit}
930   >
931     {({
932       values,
933       errors,
934       touched,
935       handleChange,
936       handleBlur,
937       handleSubmit,
938       isSubmitting,
939       isValid,
940     }) => (
941       <form onSubmit={handleSubmit}>
942         <FormField
943           htmlFor="email-input"
944           label="E-mail"
945           placeholder="email@email.com"
946           error={touched.email && errors.email}
947         >
948           <TextInput
949             id="email-input"
950             name="email"
951             type="email"
952             placeholder="email@email.com"
953             onBlur={handleBlur}
954             onChange={handleChange}
955             value={values.email}
956             required
957           />
958         </FormField>
```

```
959         <FormField
960           htmlFor="password-input"
961           label="Password"
962           error={touched.password && errors.password}
963         >
964           <TextInput
965             id="password-input"
966             name="password"
967             type="password"
968             placeholder="******"
969             onBlur={handleBlur}
970             onChange={handleChange}
971             value={values.password}
972             required
973           />
974         </FormField>
975         {authMessage && (
976           <Box margin={{ left: 'small', right: 'small' }}>
977             <Text color="status-error">{authMessage}</Text>
978           </Box>
979         )}
980         <Button
981           color="neutral-3"
982           disabled={!isValid || isSubmitting}
983           label="Log in"
984           type="submit"
985           margin={{ top: 'medium' }}
986           fill
987           primary
988         />
989       </form>
990     )}
991   </Formik>
992 );
993
994 LoginForm.propTypes = {
995   authMessage: PropTypes.string,
996   onSubmit: PropTypes.func,
997 };
998
999 LoginForm.defaultProps = {
1000   authMessage: null,
1001   onSubmit: null,
1002 };
1003
1004 const mapStateToProps = state => ({
1005   authMessage: state.auth.errorMessage,
```

```
1006 });
1007
1008 const mapDispatchToProps = {
1009   authenticate: actions.authenticate,
1010 };
1011
1012 export default compose(
1013   connect(
1014     mapStateToProps,
1015     mapDispatchToProps
1016   ),
1017   withHandlers({
1018     onSubmit: ({ authenticate }) => (values, { setSubmitting })
          => {
1019       authenticate(values);
1020       setSubmitting(false);
1021     },
1022   })
1023 )(LoginForm);
1024
1025 /**
1026  * NOME DO ARQUIVO
1027  * ./components/ServiceCallBox/index.js
1028  */
1029 /* eslint-disable no-nested-ternary */
1030 import React from 'react';
1031 import PropTypes from 'prop-types';
1032 import { withRouter } from 'react-router-dom';
1033 import { compose, withHandlers, withProps } from 'recompose';
1034 import { Button, Box, Text } from 'grommet';
1035 import { Star } from 'grommet-icons';
1036 import Rating from 'react-rating';
1037 import { format } from 'date-fns';
1038 import ms from 'pretty-ms';
1039 import { connect } from 'react-redux';
1040
1041 import NavLinkBox from 'src/styles/NavLinkBox';
1042 import { serviceCallUrl } from 'src/routes/urls';
1043 import { serviceCallCustomerSelector } from 'src/selectors/
      service-call';
1044
1045 const ServiceCallMenuItem = ({
1046   callRating,
1047   customerName,
1048   duration,
1049   isSolved,
1050   onClickAcceptCall,
```

```
1051   serviceCallId,
1052   serviceRating,
1053   startedAt,
1054 }) => (
1055   <NavLinkBox key={serviceCallId} to={serviceCallUrl(
           serviceCallId)}>
1056     <Box
1057       justify="between"
1058       height="130px"
1059       pad={{ horizontal: 'medium', vertical: 'medium' }}
1060     >
1061       <Box direction="row" justify="between" gap="medium">
1062         <Text truncate>{customerName}</Text>
1063         <Box
1064           direction="row"
1065           justify="between"
1066           gap="medium"
1067           style={{ minWidth: 'auto' }}
1068         >
1069           {!startedAt ? (
1070             <Text color="neutral-3" style={{ whiteSpace: 'nowrap'
                       }}>
1071               IS CALLING
1072             </Text>
1073           ) : (
1074             <>
1075               <Text
1076                 color={duration ? 'neutral-4' : 'neutral-5'}
1077                 style={{ whiteSpace: 'nowrap' }}
1078               >
1079                 {!duration ? 'on going' : ms(duration)}
1080               </Text>
1081               <Text
1082                 margin={{ left: 'xsmall' }}
1083                 color="neutral-4"
1084                 style={{ whiteSpace: 'nowrap' }}
1085               >
1086                 {format(startedAt, 'dd/MM')}
1087               </Text>
1088             </>
1089           )}
1090         </Box>
1091       </Box>
1092       {!startedAt ? (
1093         <Button
1094           color="accent-2"
1095           label="Answer call"
```

```
1096            onClick={onClickAcceptCall}
1097            primary
1098          />
1099        ) : (
1100          <>
1101            <div className="service-call__solved">
1102              {typeof isSolved !== 'boolean' ? (
1103                <Text color="status-warning">N/A</Text>
1104              ) : isSolved ? (
1105                <Text color="status-ok">SOLVED</Text>
1106              ) : (
1107                <Text color="status-critical">NOT SOLVED</Text>
1108              )}
1109            </div>
1110            <Box
1111              className="service-call__rating"
1112              direction="row"
1113              gap="medium"
1114              justify="between"
1115            >
1116              <Box direction="row" gap="xsmall">
1117                <Text>C:</Text>
1118                {callRating > 0 ? (
1119                  <Rating
1120                    emptySymbol={<Star />}
1121                    fullSymbol={<Star color="neutral-5" />}
1122                    initialRating={callRating}
1123                    stop={5}
1124                    readonly
1125                  />
1126                ) : (
1127                  <Text>N/A</Text>
1128                )}
1129              </Box>
1130              <Box direction="row" gap="xsmall">
1131                <Text>S:</Text>
1132                {serviceRating > 0 ? (
1133                  <Rating
1134                    emptySymbol={<Star />}
1135                    fullSymbol={<Star color="neutral-5" />}
1136                    initialRating={serviceRating}
1137                    stop={5}
1138                    readonly
1139                  />
1140                ) : (
1141                  <Text>N/A</Text>
1142                )}
```

```
1143                </Box>
1144              </Box>
1145            </>
1146          )}
1147        </Box>
1148      </NavLinkBox>
1149  );
1150
1151  ServiceCallMenuItem.propTypes = {
1152    callRating: PropTypes.number,
1153    customerName: PropTypes.string,
1154    duration: PropTypes.number,
1155    isSolved: PropTypes.bool,
1156    serviceCallId: PropTypes.string,
1157    serviceRating: PropTypes.number,
1158    startedAt: PropTypes.string,
1159    onClickAcceptCall: PropTypes.func,
1160  };
1161
1162  ServiceCallMenuItem.defaultProps = {
1163    callRating: 0,
1164    customerName: null,
1165    duration: 0,
1166    isSolved: false,
1167    serviceCallId: null,
1168    serviceRating: 0,
1169    startedAt: null,
1170    onClickAcceptCall: null,
1171  };
1172
1173  const mapStateToProps = (state, { serviceCall }) => ({
1174    customer: serviceCallCustomerSelector(state, serviceCall.
          customerId),
1175  });
1176
1177  export default compose(
1178    withRouter,
1179    connect(mapStateToProps),
1180    withProps(({ serviceCall, customer }) => ({
1181      callRating: serviceCall.callRating,
1182      customerName: customer.name || customer.email,
1183      duration: serviceCall.duration,
1184      isSolved: serviceCall.isSolved,
1185      serviceCallId: serviceCall.id,
1186      serviceRating: serviceCall.serviceRating,
1187      startedAt: serviceCall.startedAt,
1188    })),
```

```
1189   withHandlers({
1190     onClickAcceptCall: ({ serviceCallId, onAcceptCall }) => () =>
1191       onAcceptCall(serviceCallId),
1192   })
1193 )(ServiceCallMenuItem);
1194
1195 /**
1196  * NOME DO ARQUIVO
1197  * ./Root.js
1198  */
1199 import React from 'react';
1200 import { BrowserRouter as Router, Switch } from 'react-router-dom
       ';
1201 import { Provider } from 'react-redux';
1202 import { grommet } from 'grommet/themes';
1203 import { Grommet } from 'grommet';
1204
1205 import AuthProvider from 'src/containers/AuthProvider';
1206 import configureStore from 'src/store/configureStore';
1207 import Global from 'src/styles/Global';
1208 import { APP_PATH, LOGIN_PATH } from 'src/routes/paths';
1209 import AppRoute from 'src/routes/AppRoute';
1210 import LoginPage from 'src/pages/LoginPage';
1211 import RouteAuthenticated from 'src/containers/RouteAuthenticated
       ';
1212 import RouteUnauthenticated from 'src/containers/
       RouteUnauthenticated';
1213
1214 const store = configureStore();
1215
1216 const Root = () => (
1217   <Provider store={store}>
1218     <Grommet theme={grommet}>
1219       <AuthProvider>
1220         <Router>
1221           <>
1222             <Global />
1223             <Switch>
1224               <RouteUnauthenticated path={LOGIN_PATH} component={
                      LoginPage} />
1225               <RouteAuthenticated path={APP_PATH} component={
                      AppRoute} />
1226             </Switch>
1227           </>
1228         </Router>
1229       </AuthProvider>
1230     </Grommet>
```

```
1231   </Provider>
1232 );
1233
1234 export default Root;
1235
1236 /**
1237  * NOME DO ARQUIVO
1238  * ./actions/auth.js
1239  */
1240 import { RSAA } from 'redux-api-middleware';
1241
1242 import createConstants from 'src/utils/create-constants';
1243
1244 const createAuthConstants = createConstants('@@auth/');
1245
1246 export const types = createAuthConstants(
1247   // authenticate
1248   'AUTH_REQUEST',
1249   'AUTH_SUCCESS',
1250   'AUTH_FAILURE',
1251
1252   // check authentication
1253   'CHECK_AUTH_REQUEST',
1254   'CHECK_AUTH_SUCCESS',
1255   'CHECK_AUTH_FAILURE'
1256 );
1257
1258 const {
1259   AUTH_REQUEST,
1260   AUTH_SUCCESS,
1261   AUTH_FAILURE,
1262   CHECK_AUTH_REQUEST,
1263   CHECK_AUTH_SUCCESS,
1264   CHECK_AUTH_FAILURE,
1265 } = types;
1266
1267 export const authenticate = ({ email, password }) => ({
1268   [RSAA]: {
1269     endpoint: '/auth',
1270     method: 'POST',
1271     headers: {
1272       Authorization: `Basic ${btoa(`${email}:${password}`)}$`,
1273     },
1274     types: [AUTH_REQUEST, AUTH_SUCCESS, AUTH_FAILURE],
1275   },
1276 });
1277
```

```
1278 export const checkAuth = () => ({
1279   [RSAA]: {
1280     endpoint: '/users/me',
1281     method: 'GET',
1282     types: [CHECK_AUTH_REQUEST, CHECK_AUTH_SUCCESS,
         CHECK_AUTH_FAILURE],
1283   },
1284 });
1285
1286 /**
1287  * NOME DO ARQUIVO
1288  * ./actions/socket.js
1289  */
1290 import createConstants from 'src/utils/create-constants';
1291
1292 const createSocketConstants = createConstants('@@socket/');
1293
1294 export const types = createSocketConstants(
1295   'SOCKET_OPEN_REQUEST',
1296   'SOCKET_OPEN_SUCCESS',
1297   'SOCKET_CLOSE_REQUEST',
1298   'SOCKET_CLOSE_SUCCESS',
1299   'SOCKET_CONN_ERROR',
1300   'FSA',
1301   'RSSA'
1302 );
1303
1304 const { SOCKET_OPEN_REQUEST } = types;
1305
1306 export const connectSocket = () => ({
1307   type: SOCKET_OPEN_REQUEST,
1308 });
1309
1310 /**
1311  * NOME DO ARQUIVO
1312  * ./actions/customer.js
1313  */
1314 import { getJSON, RSAA } from 'redux-api-middleware';
1315 import { normalize } from 'normalizr';
1316
1317 import createConstants from 'src/utils/create-constants';
1318 import * as schema from 'src/schemas';
1319
1320 const createServiceCallConstants = createConstants('@@customers/'
       );
1321
1322 export const types = createServiceCallConstants(
```

```
1323   'ENTITY_CREATE',
1324   'ENTITY_DELETE',
1325   // READ ENTITIES
1326   'ENTITIES_READ_REQUEST',
1327   'ENTITIES_READ_SUCCESS',
1328   'ENTITIES_READ_FAILURE'
1329 );
1330
1331 const {
1332   ENTITIES_READ_REQUEST,
1333   ENTITIES_READ_SUCCESS,
1334   ENTITIES_READ_FAILURE,
1335 } = types;
1336
1337 export const readServiceCalls = () => ({
1338   [RSAA]: {
1339     endpoint: '/service-calls',
1340     method: 'GET',
1341     types: [
1342       ENTITIES_READ_REQUEST,
1343       {
1344         type: ENTITIES_READ_SUCCESS,
1345         payload: (action, state, res) =>
1346           getJSON(res).then(json => normalize(json, [schema.
                serviceCall])),
1347       },
1348       ENTITIES_READ_FAILURE,
1349     ],
1350   },
1351 });
1352
1353 /**
1354  * NOME DO ARQUIVO
1355  * ./actions/__tests__/auth.js
1356  */
1357 import configureMockStore from 'redux-mock-store';
1358 import { apiMiddleware } from 'redux-api-middleware';
1359 import thunk from 'redux-thunk';
1360 import fetchMock from 'fetch-mock';
1361
1362 import * as actions from 'src/actions/auth';
1363
1364 const middlewares = [thunk, apiMiddleware];
1365 const mockStore = configureMockStore(middlewares);
1366
1367 describe('[actions] authenticate', () => {
1368   afterEach(() => {
```

```
1369      fetchMock.reset();
1370      fetchMock.restore();
1371    });
1372
1373    it('should dispatch AUTH_REQUEST and AUTH_SUCCESS when
          authenticate is called', () => {
1374      const store = mockStore({});
1375      const user = {
1376        email: 'admin@sac.com',
1377        password: '123456',
1378      };
1379
1380      fetchMock.postOnce('/auth', {
1381        body: user,
1382        headers: { 'content-type': 'application/json' },
1383      });
1384
1385      const expectedActions = [
1386        { type: actions.AUTH_REQUEST },
1387        { type: actions.AUTH_SUCCESS, payload: user },
1388      ];
1389      return store.dispatch(actions.authenticate({ ...user })).then
            (() => {
1390        expect(store.getActions()).toEqual(expectedActions);
1391      });
1392    });
1393  });
1394
1395  /**
1396   * NOME DO ARQUIVO
1397   * ./actions/service-call.js
1398   */
1399  import { getJSON, RSAA } from 'redux-api-middleware';
1400  import { normalize } from 'normalizr';
1401
1402  import createConstants from 'src/utils/create-constants';
1403  import * as schema from 'src/schemas';
1404
1405  import { types as socketTypes } from './socket';
1406
1407  const createServiceCallConstants = createConstants('@@service-
        call/');
1408
1409  export const types = createServiceCallConstants(
1410    'ENTITY_CREATE',
1411    'ENTITY_DELETE',
1412    'ENTITY_UPDATE',
```

```
1413
1414   // READ ENTITIES
1415   'ENTITIES_READ_REQUEST',
1416   'ENTITIES_READ_SUCCESS',
1417   'ENTITIES_READ_FAILURE'
1418 );
1419
1420 const {
1421   ENTITY_UPDATE,
1422
1423   ENTITIES_READ_REQUEST,
1424   ENTITIES_READ_SUCCESS,
1425   ENTITIES_READ_FAILURE,
1426 } = types;
1427
1428 export const readServiceCalls = () => ({
1429   [RSAA]: {
1430     endpoint: `/service-calls?filter=${JSON.stringify({
1431       eager: {
1432         customer: true,
1433       },
1434       order: 'startedAt desc, updatedAt desc',
1435     })}`,
1436     method: 'GET',
1437     types: [
1438       ENTITIES_READ_REQUEST,
1439       {
1440         type: ENTITIES_READ_SUCCESS,
1441         payload: (action, state, res) =>
1442           getJSON(res).then(json => normalize(json, [schema.
                serviceCall])),
1443       },
1444       ENTITIES_READ_FAILURE,
1445     ],
1446   },
1447 });
1448
1449 export const updateServiceCall = data => ({
1450   [socketTypes.RSSA]: {
1451     event: ENTITY_UPDATE,
1452     message: { data },
1453   },
1454 });
1455
1456 /**
1457  * NOME DO ARQUIVO
1458  * ./actions/rtc.js
```

```
1459  */
1460 import { types as socketTypes } from 'src/actions/socket';
1461 import createConstants from 'src/utils/create-constants';
1462
1463 const createRtcConstants = createConstants('@@rtc/');
1464 const { RSSA } = socketTypes;
1465
1466 export const types = createRtcConstants(
1467   'PEER_CONNECT',
1468   'PEER_DISCONNECT',
1469   'PEER_SET',
1470   'STREAM_SET',
1471   'SIGNAL_RECEIVE',
1472   'SIGNAL_SEND'
1473 );
1474
1475 const {
1476   PEER_CONNECT,
1477   PEER_DISCONNECT,
1478   PEER_SET,
1479   STREAM_SET,
1480   SIGNAL_RECEIVE,
1481   SIGNAL_SEND,
1482 } = types;
1483
1484 export const sendSignal = ({ room, signal }) => ({
1485   [RSSA]: {
1486     event: SIGNAL_SEND,
1487     message: {
1488       type: SIGNAL_RECEIVE,
1489       payload: { signal },
1490       meta: { namespace: '/caller', room },
1491     },
1492   },
1493 });
1494
1495 export const connectPeer = ({ room, stream }) => ({
1496   [RSSA]: {
1497     event: PEER_CONNECT,
1498     message: {
1499       type: PEER_CONNECT,
1500       payload: { stream },
1501       meta: { namespace: '/caller', room },
1502     },
1503     optimistic: true,
1504   },
1505 });
```

```
1506
1507 export const disconnectPeer = ({ err } = {}) => ({
1508   type: PEER_DISCONNECT,
1509   payload: { err },
1510 });
1511
1512 export const receiveSignal = signal => ({
1513   type: SIGNAL_RECEIVE,
1514   payload: { signal },
1515 });
1516
1517 export const setPeer = peer => ({
1518   type: PEER_SET,
1519   payload: { peer },
1520 });
1521
1522 export const setStream = stream => ({
1523   type: STREAM_SET,
1524   payload: { stream },
1525 });
1526
1527 /**
1528  * NOME DO ARQUIVO
1529  * ./schemas.js
1530  */
1531 import { schema } from 'normalizr';
1532
1533 export const customer = new schema.Entity('customers');
1534 export const serviceCall = new schema.Entity('serviceCalls', {
1535   customer,
1536 });
1537
1538 /**
1539  * NOME DO ARQUIVO
1540  * ./containers/RouteUnauthenticated/index.js
1541  */
1542 import React from 'react';
1543 import PropTypes from 'prop-types';
1544 import { connect } from 'react-redux';
1545 import { Route, Redirect } from 'react-router-dom';
1546
1547 import { DASHBOARD_PATH } from 'src/routes/paths';
1548 import { isAuthenticatedSelector } from 'src/selectors/auth';
1549
1550 const RouteUnauthenticated = ({
1551   component: Component,
1552   isAuthenticated,
```

```
1553    ...rest
1554  }) => {
1555    if (isAuthenticated) {
1556      return <Redirect to={DASHBOARD_PATH} {...rest} />;
1557    }
1558
1559    return <Route {...rest} component={Component} />;
1560  };
1561
1562  RouteUnauthenticated.propTypes = {
1563    component: PropTypes.func.isRequired,
1564    isAuthenticated: PropTypes.bool.isRequired,
1565  };
1566
1567  const mapStateToProps = state => ({
1568    isAuthenticated: isAuthenticatedSelector(state),
1569  });
1570
1571  export default connect(mapStateToProps)(RouteUnauthenticated);
1572
1573  /**
1574   * NOME DO ARQUIVO
1575   * ./containers/RouteAuthenticated/index.js
1576   */
1577  import React from 'react';
1578  import PropTypes from 'prop-types';
1579  import { connect } from 'react-redux';
1580  import { Route, Redirect } from 'react-router-dom';
1581
1582  import { LOGIN_PATH } from 'src/routes/paths';
1583  import { isAuthenticatedSelector } from 'src/selectors/auth';
1584
1585  const RouteAuthenticated = ({
1586    component: Component,
1587    isAuthenticated,
1588    ...rest
1589  }) => {
1590    if (!isAuthenticated) {
1591      return <Redirect to={LOGIN_PATH} {...rest} />;
1592    }
1593
1594    return <Route {...rest} component={Component} />;
1595  };
1596
1597  RouteAuthenticated.propTypes = {
1598    component: PropTypes.func.isRequired,
1599    isAuthenticated: PropTypes.bool.isRequired,
```

```
1600 };
1601
1602 const mapStateToProps = state => ({
1603   isAuthenticated: isAuthenticatedSelector(state),
1604 });
1605
1606 export default connect(mapStateToProps)(RouteAuthenticated);
1607
1608 /**
1609  * NOME DO ARQUIVO
1610  * ./containers/VideoChat/index.js
1611  */
1612 import { connect } from 'react-redux';
1613 import { compose, lifecycle, withHandlers } from 'recompose';
1614
1615 import { CONNECTION_STATE as RTC_CONNECTION_STATE } from 'src/
        reducers/rtc';
1616 import * as rtcActions from 'src/actions/rtc';
1617 import * as serviceCallActions from 'src/actions/service-call';
1618
1619 import VideoChat from './components/VideoChat';
1620
1621 const rtcConnectionStateSelector = state => state.rtc.
        connectionState;
1622
1623 const mapStateToProps = state => ({
1624   localStream: state.rtc.localStream,
1625   remoteStream: state.rtc.remoteStream,
1626   rtcConnectionState: rtcConnectionStateSelector(state),
1627   user: state.auth.user,
1628 });
1629
1630 const mapDispatchToProps = {
1631   connectPeer: rtcActions.connectPeer,
1632   disconnectPeer: rtcActions.disconnectPeer,
1633   updateServiceCall: serviceCallActions.updateServiceCall,
1634 };
1635
1636 export default compose(
1637   connect(
1638     mapStateToProps,
1639     mapDispatchToProps
1640   ),
1641   withHandlers({
1642     onClickHangUp: ({ disconnectPeer, localStream }) => () => {
1643       if (localStream) {
1644         localStream.getTracks().forEach(track => track.stop());
```

```
1645        }
1646        disconnectPeer();
1647      },
1648    }),
1649    lifecycle({
1650      componentDidMount() {
1651        const { room, connectPeer } = this.props;
1652
1653        navigator.mediaDevices
1654          .getUserMedia({
1655            video: {
1656              width: { max: 1920 },
1657              height: { max: 1920 },
1658            },
1659            audio: true,
1660          })
1661          .then(stream => {
1662            connectPeer({ room, stream });
1663          });
1664      },
1665      componentDidUpdate(prevProps) {
1666        const {
1667          onHangUp,
1668          rtcConnectionState,
1669          room,
1670          user,
1671          updateServiceCall,
1672        } = this.props;
1673
1674        if (
1675          rtcConnectionState !== prevProps.rtcConnectionState &&
1676          rtcConnectionState === RTC_CONNECTION_STATE.CONNECTED
1677        ) {
1678          updateServiceCall({ id: room, userId: user.id, startedAt:
                 new Date() });
1679        }
1680
1681        if (
1682          prevProps.rtcConnectionState === RTC_CONNECTION_STATE.
                CONNECTED &&
1683          rtcConnectionState === RTC_CONNECTION_STATE.DISCONNECTED
1684        ) {
1685          updateServiceCall({ id: room, endedAt: new Date() });
1686          onHangUp();
1687        }
1688      },
1689    })
```

```
)(VideoChat);

/**
 * NOME DO ARQUIVO
 * ./containers/VideoChat/styles/VideoChatOverlay.js
 */
import styled, { keyframes } from 'styled-components';
import { Box } from 'grommet';

const fadeOut = keyframes`
  0% {
    opacity: 1;
  }
  100% {
    opacity: 0;
  }
`;

const VideoChatOverlay = styled(Box).attrs({
  round: true,
})`
  animation: 0.3s ${fadeOut} 3.2s ease-in forwards;
  background-color: rgba(221, 221, 221, 0.2);
  height: 100%;
  left: 0;
  overflow: hidden;
  position: absolute;
  top: 0;
  width: 100%;

  &:active,
  &:hover {
    animation: unset;
    opacity: 1;
    transition: unset;
  }
`;

export default VideoChatOverlay;

/**
 * NOME DO ARQUIVO
 * ./containers/VideoChat/styles/VideoBox.js
 */
import styled from 'styled-components';
import { Box } from 'grommet';

```

```
1737 const VideoBox = styled(Box)`
1738   position: absolute;
1739   top: 12px;
1740   right: 12px;
1741   width: 50%;
1742
1743   @media screen and (min-width: 36em) {
1744     width: 25%;
1745   }
1746 `;
1747
1748 export default VideoBox;
1749
1750 /**
1751  * NOME DO ARQUIVO
1752  * ./containers/VideoChat/components/VideoChat.js
1753  */
1754 import React, { Component, createRef } from 'react';
1755 import PropTypes from 'prop-types';
1756 import { Button, Box, Layer } from 'grommet';
1757 import { Stretch } from 'styled-loaders-react';
1758
1759 import VideoBox from '../styles/VideoBox';
1760 import VideoChatOverlay from '../styles/VideoChatOverlay';
1761
1762 class VideoChat extends Component {
1763   constructor(props) {
1764     super(props);
1765
1766     this.localVideoRef = createRef();
1767     this.remoteVideoRef = createRef();
1768   }
1769
1770   componentDidUpdate() {
1771     const { localStream, remoteStream } = this.props;
1772
1773     if (localStream) {
1774       this.localVideoRef.current.srcObject = localStream;
1775     }
1776
1777     if (remoteStream) {
1778       this.remoteVideoRef.current.srcObject = remoteStream;
1779     }
1780   }
1781
1782   render() {
```

```
1783    const { localStream, remoteStream, onClickHangUp } = this.
            props;
1784    return (
1785      <Layer
1786        position="center"
1787        margin="large"
1788        style={{
1789          backgroundColor: 'transparent',
1790          height: '100%',
1791          overflow: 'hidden',
1792        }}
1793        modal
1794        full
1795      >
1796        <Box
1797          align="center"
1798          justify="center"
1799          background="#000"
1800          overflow="hidden"
1801          elevation="xlarge"
1802          round
1803          fill
1804        >
1805          {remoteStream ? (
1806            <video
1807              style={{
1808                width: '100%',
1809                height: '100%',
1810                objectFit: 'contain',
1811                transform: 'scaleX(-1)',
1812              }}
1813              ref={this.remoteVideoRef}
1814              autoPlay
1815              muted
1816            />
1817          ) : (
1818            <Stretch color="#7D4CDB" size="100px" />
1819          )}
1820          <VideoChatOverlay>
1821            <VideoBox
1822              align="center"
1823              justify="center"
1824              background="#000"
1825              elevation="xlarge"
1826              overflow="hidden"
1827              round
1828            >
```

```
1829                {localStream ? (
1830                  <video
1831                    style={{
1832                      width: '100%',
1833                      height: '100%',
1834                      objectFit: 'contain',
1835                      transform: 'scaleX(-1)',
1836                    }}
1837                    ref={this.localVideoRef}
1838                    autoPlay
1839                    muted
1840                  />
1841                ) : (
1842                  <Stretch color="#7D4CDB" size="100px" />
1843                )}
1844              </VideoBox>
1845              <Button
1846                color="status-critical"
1847                label="Hang up"
1848                style={{
1849                  position: 'absolute',
1850                  bottom: '24px',
1851                  left: '50%',
1852                  transform: 'translateX(-50%)',
1853                }}
1854                onClick={onClickHangUp}
1855                primary
1856              />
1857            </VideoChatOverlay>
1858          </Box>
1859        </Layer>
1860      );
1861    }
1862  }
1863
1864  VideoChat.propTypes = {
1865    localStream: PropTypes.shape({
1866      id: PropTypes.string,
1867    }),
1868    remoteStream: PropTypes.shape({
1869      id: PropTypes.string,
1870    }),
1871    onClickHangUp: PropTypes.func,
1872  };
1873
1874  VideoChat.defaultProps = {
1875    localStream: null,
```

```
1876   remoteStream: null,
1877   onClickHangUp: null,
1878 };
1879
1880 export default VideoChat;
1881
1882 /**
1883  * NOME DO ARQUIVO
1884  * ./containers/AuthProvider/index.js
1885  */
1886 import { Children, Component } from 'react';
1887 import PropTypes from 'prop-types';
1888 import { connect } from 'react-redux';
1889
1890 import * as authActions from 'src/actions/auth';
1891 import * as socketActions from 'src/actions/socket';
1892
1893 class AuthProvider extends Component {
1894   componentDidMount() {
1895     const { checkAuth, connectSocket, isAuthenticated } = this.
          props;
1896
1897     checkAuth();
1898
1899     if (isAuthenticated) {
1900       connectSocket();
1901     }
1902   }
1903
1904   componentDidUpdate() {
1905     const { connectSocket, isAuthenticated } = this.props;
1906
1907     if (isAuthenticated) {
1908       connectSocket();
1909     }
1910   }
1911
1912   render() {
1913     const { children } = this.props;
1914     return Children.only(children);
1915   }
1916 }
1917
1918 AuthProvider.propTypes = {
1919   checkAuth: PropTypes.func.isRequired,
1920   children: PropTypes.node.isRequired,
1921   connectSocket: PropTypes.func.isRequired,
```

```
1922   isAuthenticated: PropTypes.bool.isRequired,
1923 };
1924
1925 const mapStateToProps = state => ({
1926   isAuthenticated: state.auth.isAuthenticated,
1927 });
1928
1929 const mapDispatchToProps = {
1930   checkAuth: authActions.checkAuth,
1931   connectSocket: socketActions.connectSocket,
1932 };
1933
1934 export default connect(
1935   mapStateToProps,
1936   mapDispatchToProps
1937 )(AuthProvider);
1938
1939 /**
1940  * NOME DO ARQUIVO
1941  * ./setupTests.js
1942  */
1943 import 'jest-styled-components';
1944
1945 /**
1946  * NOME DO ARQUIVO
1947  * ./pages/ServiceCallPage/index.js
1948  */
1949 /* eslint-disable no-nested-ternary */
1950 import React from 'react';
1951 import PropTypes from 'prop-types';
1952 import { Flex } from '@rebass/grid';
1953 import { Box, DataTable, Stack, Heading, Text } from 'grommet';
1954 import Rating from 'react-rating';
1955 import { Star } from 'grommet-icons';
1956
1957 const ServiceCallPage = ({
1958   callRating,
1959   customerEmail,
1960   customerName,
1961   description,
1962   duration,
1963   endedAt,
1964   isSolved,
1965   serviceRating,
1966   startedAt,
1967 }) => (
1968   <Flex
```

```
    flexDirection="column"
    justifyContent="center"
    css={{ height: '100%', maxWidth: '560px', margin: 'auto' }}
    width={1 / 2}
  >
    <Box>
      <Heading level="2">Service call</Heading>
      <Stack guidingChild="last">
        <DataTable
          columns={[
            {
              property: 'key',
              primary: true,
            },
            {
              property: 'value',
            },
          ]}
          data={[
            {
              key: 'Start / end / duration',
              value: `${startedAt} / ${endedAt} / ${duration}`,
            },
            {
              key: 'Call rating',
              value: (
                <Box pad={{ vertical: 'xsmall' }}>
                  <Rating
                    emptySymbol={<Star />}
                    fullSymbol={<Star color="neutral-5" />}
                    initialRating={callRating}
                    stop={5}
                    readonly
                  />
                </Box>
              ),
            },
            {
              key: 'Service rating',
              value: (
                <Box pad={{ vertical: 'xsmall' }}>
                  <Rating
                    emptySymbol={<Star />}
                    fullSymbol={<Star color="neutral-5" />}
                    initialRating={serviceRating}
                    stop={5}
                    readonly
```

```
                    />
                  </Box>
                ),
              },
              {
                key: 'Is solved',
                value:
                  typeof isSolved !== 'boolean' ? (
                    <Text color="neutral-5" weight="bold">
                      N/A
                    </Text>
                  ) : isSolved ? (
                    <Text color="neutral-3" weight="bold">
                      SOLVED
                    </Text>
                  ) : (
                    <Text color="status-critical" weight="bold">
                      NOT SOLVED
                    </Text>
                  ),
              },
              {
                key: 'Description',
                value: description,
              },
            ]}
          />
        </Stack>
      </Box>
      <Box margin={{ top: 'medium' }}>
        <Heading level="2">Customer</Heading>
        <Stack guidingChild="last">
          <DataTable
            columns={[
              {
                property: 'key',
                primary: true,
              },
              {
                property: 'value',
              },
            ]}
            data={[
              {
                key: 'Name',
                value: customerName,
              },
```

```
2063                {
2064                  key: 'E-mail',
2065                  value: customerEmail,
2066                },
2067              ]}
2068            />
2069          </Stack>
2070        </Box>
2071      </Flex>
2072    );
2073
2074    ServiceCallPage.propTypes = {
2075      callRating: PropTypes.number,
2076      customerEmail: PropTypes.string,
2077      customerName: PropTypes.string,
2078      description: PropTypes.string,
2079      duration: PropTypes.string,
2080      endedAt: PropTypes.string,
2081      isSolved: PropTypes.bool,
2082      serviceRating: PropTypes.number,
2083      startedAt: PropTypes.string,
2084    };
2085
2086    ServiceCallPage.defaultProps = {
2087      callRating: null,
2088      customerEmail: null,
2089      customerName: null,
2090      description: null,
2091      duration: null,
2092      endedAt: null,
2093      isSolved: null,
2094      serviceRating: null,
2095      startedAt: null,
2096    };
2097
2098    export default ServiceCallPage;
2099
2100    /**
2101     * NOME DO ARQUIVO
2102     * ./pages/LoginPage/index.js
2103     */
2104    import React from 'react';
2105    import { Box, Grommet } from 'grommet';
2106
2107    import LoginForm from 'src/components/LoginForm';
2108
2109    const LoginPage = () => (
```

```
2110   <Grommet full>
2111     <Box
2112       align="center"
2113       alignContent="center"
2114       alignSelf="center"
2115       justify="center"
2116       fill
2117     >
2118       <Box
2119         width="medium"
2120         border={{ color: 'neutral-1', size: 'large' }}
2121         pad="large"
2122       >
2123         <LoginForm />
2124       </Box>
2125     </Box>
2126   </Grommet>
2127 );
2128
2129 export default LoginPage;
2130
2131 /**
2132  * NOME DO ARQUIVO
2133  * ./pages/DashboardPage/index.js
2134  */
2135 import React from 'react';
2136 import PropTypes from 'prop-types';
2137 import { Box as GridBox, Flex } from '@rebass/grid';
2138 import { Box, DataTable, Heading, Stack, Meter, Text } from '
       grommet';
2139 import InlineFlex from 'src/styles/InlineFlex';
2140
2141 const percentage = (value, total) => (value / total) * 100;
2142
2143 const DashboardPage = ({
2144   avgCallRating,
2145   avgServiceRating,
2146   solvedCalls,
2147   notSolvedCalls,
2148   notAnsweredCalls,
2149   totalCalls,
2150 }) => (
2151   <Flex
2152     flexDirection="column"
2153     justifyContent="center"
2154     css={{ height: '100%', maxWidth: '480px', margin: 'auto' }}
2155     width={1 / 2}
```

```
2156  >
2157    <GridBox>
2158      <InlineFlex
2159        alignItems="center"
2160        flexDirection="column"
2161        width={[1, null, null, 1 / 2]}
2162        p={['12px', null, null, '24px']}
2163      >
2164        <Heading level="4" margin={{ bottom: 'medium' }}>
2165          CALL RATING AVG
2166        </Heading>
2167        <Box align="start">
2168          <Stack anchor="center">
2169            <Meter
2170              type="circle"
2171              background="light-1"
2172              values={[
2173                {
2174                  value: (avgCallRating / 5) * 100,
2175                  color: 'brand',
2176                },
2177              ]}
2178              size="small"
2179              thickness="medium"
2180            />
2181            <Box direction="row" align="center" pad={{ bottom: '
                    xsmall' }}>
2182              <Text size="xlarge" weight="bold">
2183                {avgCallRating}
2184              </Text>
2185              <Text size="small"> /5</Text>
2186            </Box>
2187          </Stack>
2188        </Box>
2189      </InlineFlex>
2190      <InlineFlex
2191        alignItems="center"
2192        flexDirection="column"
2193        width={[1, null, null, 1 / 2]}
2194        p="24px"
2195      >
2196        <Heading level="4" margin={{ bottom: 'medium' }}>
2197          SERVICE RATING AVG
2198        </Heading>
2199        <Box align="start">
2200          <Stack anchor="center">
2201            <Meter
```

```
2202                type="circle"
2203                background="light-1"
2204                values={[
2205                  {
2206                    value: (avgServiceRating / 5) * 100,
2207                    color: 'brand',
2208                  },
2209                ]}
2210                size="small"
2211                thickness="medium"
2212              />
2213              <Box direction="row" align="center" pad={{ bottom: '
                      xsmall' }}>
2214                <Text size="xlarge" weight="bold">
2215                  {avgServiceRating}
2216                </Text>
2217                <Text size="small"> /5</Text>
2218              </Box>
2219            </Stack>
2220          </Box>
2221        </InlineFlex>
2222      </GridBox>
2223      <Flex alignItems="center" flexDirection="column" width={1} p=
          "24px">
2224        <Heading level="4" margin={{ bottom: 'medium' }}>
2225          SOLVED STATISTICS
2226        </Heading>
2227        <Box align="center">
2228          <Stack guidingChild="last">
2229            <DataTable
2230              columns={[
2231                {
2232                  property: 'key',
2233                  primary: true,
2234                },
2235                {
2236                  property: 'value',
2237                },
2238                {
2239                  property: 'percentage',
2240                  render: datum => (
2241                    <Box pad={{ vertical: 'xsmall' }}>
2242                      <Meter
2243                        background="light-1"
2244                        values={[{ color: 'brand', value: datum.
                              percentage }]}
2245                        thickness="small"
```

```
2246                         size="small"
2247                       />
2248                     </Box>
2249                   ),
2250                 },
2251               ]}
2252               data={[
2253                 {
2254                   key: 'Solved',
2255                   value: solvedCalls,
2256                   percentage: percentage(solvedCalls, totalCalls),
2257                 },
2258                 {
2259                   key: 'Not solved',
2260                   value: notSolvedCalls,
2261                   percentage: percentage(notSolvedCalls, totalCalls
                        ),
2262                 },
2263                 {
2264                   key: 'N/A',
2265                   value: notAnsweredCalls,
2266                   percentage: percentage(notAnsweredCalls,
                        totalCalls),
2267                 },
2268                 {
2269                   key: 'Total',
2270                   value: totalCalls,
2271                   percentage: percentage(totalCalls, totalCalls),
2272                 },
2273               ]}
2274             />
2275           </Stack>
2276         </Box>
2277       </Flex>
2278     </Flex>
2279 );
2280
2281 DashboardPage.propTypes = {
2282   avgCallRating: PropTypes.string,
2283   avgServiceRating: PropTypes.string,
2284   notAnsweredCalls: PropTypes.number,
2285   notSolvedCalls: PropTypes.number,
2286   solvedCalls: PropTypes.number,
2287   totalCalls: PropTypes.number,
2288 };
2289
2290 DashboardPage.defaultProps = {
```

```
2291    avgCallRating: 0,
2292    avgServiceRating: 0,
2293    notAnsweredCalls: 0,
2294    notSolvedCalls: 0,
2295    solvedCalls: 0,
2296    totalCalls: 0,
2297  };
2298
2299  export default DashboardPage;
2300
2301  /**
2302   * NOME DO ARQUIVO
2303   * ./pages/AppPage/index.js
2304   */
2305  import React from 'react';
2306  import PropTypes from 'prop-types';
2307  import { Redirect, Route, Switch } from 'react-router-dom';
2308  import { Box, Text, Grommet, Grid } from 'grommet';
2309  import { Dashboard } from 'grommet-icons';
2310
2311  import { dashboardUrl } from 'src/routes/urls';
2312  import NavLinkBox from 'src/styles/NavLinkBox';
2313  import { DASHBOARD_PATH, SERVICE_CALL_PATH } from 'src/routes/
       paths';
2314  import ServiceCallBox from 'src/components/ServiceCallBox';
2315  import DashboardRoute from 'src/routes/DashboardRoute';
2316  import ServiceCallRoute from 'src/routes/ServiceCallRoute';
2317  import VideoChat from 'src/containers/VideoChat';
2318
2319  const AppPage = ({ serviceCallRoom, serviceCalls,
       setServiceCallRoom }) => (
2320    <Grommet full>
2321      <Grid
2322        areas={[
2323          { name: 'sidebar', start: [0, 0], end: [0, 0] },
2324          { name: 'main', start: [1, 0], end: [1, 0] },
2325        ]}
2326        columns={['medium', 'flex']}
2327        rows={['flex']}
2328        fill
2329      >
2330        <Box
2331          gridArea="sidebar"
2332          background="dark-1"
2333          width="medium"
2334          overflow={{ horizontal: 'auto' }}
2335        >
```

```
2336          <NavLinkBox to={dashboardUrl()} exact>
2337            <Box
2338              direction="row"
2339              gap="small"
2340              pad={{ horizontal: 'medium', vertical: 'medium' }}
2341            >
2342              <Dashboard />
2343              <Text>Dashboard</Text>
2344            </Box>
2345          </NavLinkBox>
2346          {serviceCalls.map(serviceCall => (
2347            <ServiceCallBox
2348              key={serviceCall.id}
2349              serviceCall={serviceCall}
2350              onAcceptCall={serviceCallId => setServiceCallRoom(
                    serviceCallId)}
2351            />
2352          ))}
2353        </Box>
2354        <Box
2355          background="light-3"
2356          gridArea="main"
2357          overflow={{ horizontal: 'auto' }}
2358        >
2359          <Switch>
2360            <Route path={DASHBOARD_PATH} component={DashboardRoute}
                    />
2361            <Route path={SERVICE_CALL_PATH} component={
                  ServiceCallRoute} />
2362            <Route render={() => <Redirect to={dashboardUrl()} />}
                  />
2363          </Switch>
2364        </Box>
2365      </Grid>
2366      {serviceCallRoom && (
2367        <VideoChat
2368          room={serviceCallRoom}
2369          onHangUp={() => setServiceCallRoom(null)}
2370        />
2371      )}
2372    </Grommet>
2373  );
2374
2375  AppPage.propTypes = {
2376    serviceCallRoom: PropTypes.string,
2377    serviceCalls: PropTypes.arrayOf(
2378      PropTypes.shape({
```

```
2379        id: PropTypes.string,
2380      })
2381    ),
2382    setServiceCallRoom: PropTypes.func,
2383  };
2384
2385  AppPage.defaultProps = {
2386    serviceCallRoom: null,
2387    serviceCalls: null,
2388    setServiceCallRoom: null,
2389  };
2390
2391  export default AppPage;
2392
2393  /**
2394   * NOME DO ARQUIVO
2395   * ./pages/LoadingPage/index.js
2396   */
2397  import React from 'react';
2398  import PropTypes from 'prop-types';
2399  import { Grommet, Box } from 'grommet';
2400  import { Stretch } from 'styled-loaders-react';
2401
2402  const LoadingPage = ({ full }) =>
2403    full ? (
2404      <Grommet full>
2405        <Box align="center" background="dark-1" justify="center"
               fill>
2406          <Stretch color="#7D4CDB" size="100px" />
2407        </Box>
2408      </Grommet>
2409    ) : (
2410      <Box fill>
2411        <Box align="center" background="dark-1" justify="center"
               fill>
2412          <Stretch color="#7D4CDB" size="100px" />
2413        </Box>
2414      </Box>
2415    );
2416
2417  LoadingPage.propTypes = {
2418    full: PropTypes.bool,
2419  };
2420
2421  LoadingPage.defaultProps = {
2422    full: false,
2423  };
```

```
2424
2425 export default LoadingPage;
2426
2427 /**
2428  * NOME DO ARQUIVO
2429  * ./routes/ServiceCallRoute/index.js
2430  */
2431 import { connect } from 'react-redux';
2432 import { compose, branch, renderComponent, withProps } from '
        recompose';
2433 import { format } from 'date-fns';
2434 import ms from 'pretty-ms';
2435
2436 import LoadingPage from 'src/pages/LoadingPage';
2437 import ServiceCallPage from 'src/pages/ServiceCallPage';
2438 import {
2439   serviceCallCustomerSelector,
2440   serviceCallSelector,
2441 } from 'src/selectors/service-call';
2442
2443 const mapStateToProps = (state, { match }) => {
2444   const serviceCall = serviceCallSelector(state, match.params.
          serviceCallId);
2445   let customer;
2446
2447   if (serviceCall) {
2448     customer = serviceCallCustomerSelector(state, serviceCall.
            customerId);
2449   }
2450
2451   return {
2452     customer,
2453     serviceCall,
2454   };
2455 };
2456
2457 export default compose(
2458   connect(mapStateToProps),
2459   branch(({ serviceCall }) => !serviceCall, renderComponent(
          LoadingPage)),
2460   withProps(({ customer, serviceCall }) => {
2461     const {
2462       callRating,
2463       description,
2464       duration,
2465       endedAt,
2466       isSolved,
```

```
2467        serviceRating,
2468        startedAt,
2469      } = serviceCall;
2470      const { email, name } = customer;
2471
2472      return {
2473        callRating,
2474        customerEmail: email,
2475        customerName: name,
2476        description,
2477        duration: duration ? ms(duration) : 'on going',
2478        endedAt: format(endedAt, 'HH:mm dd/MM'),
2479        isSolved,
2480        serviceRating,
2481        startedAt: format(startedAt, 'HH:mm dd/MM'),
2482      };
2483    })
2484  )(ServiceCallPage);
2485
2486  /**
2487   * NOME DO ARQUIVO
2488   * ./routes/DashboardRoute/index.js
2489   */
2490  import { connect } from 'react-redux';
2491
2492  import DashboardPage from 'src/pages/DashboardPage';
2493  import {
2494    avgCallRatingSelector,
2495    avgServiceRatingSelector,
2496    solvedStatisticsSelector,
2497  } from 'src/selectors/service-call';
2498
2499  const mapStateToProps = state => {
2500    const {
2501      solvedCalls,
2502      notSolvedCalls,
2503      notAnsweredCalls,
2504    } = solvedStatisticsSelector(state);
2505
2506    return {
2507      avgCallRating: avgCallRatingSelector(state).toFixed(2),
2508      avgServiceRating: avgServiceRatingSelector(state).toFixed(2),
2509      solvedCalls,
2510      notSolvedCalls,
2511      notAnsweredCalls,
2512      totalCalls: solvedCalls + notSolvedCalls + notAnsweredCalls,
2513    };
```

```
2514 };
2515
2516 export default connect(mapStateToProps)(DashboardPage);
2517
2518 /**
2519  * NOME DO ARQUIVO
2520  * ./routes/paths.js
2521  */
2522 export const APP_PATH = '';
2523 export const DASHBOARD_PATH = `${APP_PATH}/dashboard`;
2524 export const LOGIN_PATH = `${APP_PATH}/login`;
2525 export const SERVICE_CALL_PATH = `${APP_PATH}/:serviceCallId`;
2526
2527 /**
2528  * NOME DO ARQUIVO
2529  * ./routes/urls.js
2530  */
2531 import {
2532   APP_PATH,
2533   DASHBOARD_PATH,
2534   LOGIN_PATH,
2535   SERVICE_CALL_PATH,
2536 } from './paths';
2537
2538 export const appUrl = () => APP_PATH;
2539 export const dashboardUrl = () => DASHBOARD_PATH;
2540 export const loginUrl = () => LOGIN_PATH;
2541
2542 export const serviceCallUrl = serviceCallId =>
2543   SERVICE_CALL_PATH.replace(':serviceCallId', serviceCallId);
2544
2545 /**
2546  * NOME DO ARQUIVO
2547  * ./routes/AppRoute/index.js
2548  */
2549 import { compose, withState } from 'recompose';
2550 import { connect } from 'react-redux';
2551
2552 import withDataLoading from 'src/hocs/with-data-loading';
2553 import AppPage from 'src/pages/AppPage';
2554 import * as serviceCallsActions from 'src/actions/service-call';
2555 import { serviceCallsSelector } from 'src/selectors/service-call'
      ;
2556
2557 const mapStateToProps = state => ({
2558   serviceCalls: serviceCallsSelector(state),
2559 });
```

```
2560
2561 const mapDispatchToProps = {
2562   readServiceCalls: serviceCallsActions.readServiceCalls,
2563 };
2564
2565 const loadData = props => props.readServiceCalls();
2566
2567 export default compose(
2568   connect(
2569     mapStateToProps,
2570     mapDispatchToProps
2571   ),
2572   withDataLoading(loadData),
2573   withState('serviceCallRoom', 'setServiceCallRoom', null)
2574 )(AppPage);
2575
2576 /**
2577  * NOME DO ARQUIVO
2578  * ./reducers/auth.js
2579  */
2580 import produce from 'immer';
2581
2582 import { types } from 'src/actions/auth';
2583 import els from 'src/utils/expirable-local-storage';
2584 import { AUTH_TOKEN_KEY } from 'src/constants';
2585
2586 const {
2587   AUTH_FAILURE,
2588   AUTH_REQUEST,
2589   AUTH_SUCCESS,
2590   CHECK_AUTH_REQUEST,
2591   CHECK_AUTH_SUCCESS,
2592   CHECK_AUTH_FAILURE,
2593 } = types;
2594
2595 export const STATE_KEY = 'auth';
2596
2597 export const state = {
2598   errorMessage: null,
2599   isAuthenticated: !!els.get(AUTH_TOKEN_KEY),
2600   isFetching: false,
2601   user: null,
2602 };
2603
2604 const reducer = produce((draft, { type, payload }) => {
2605   switch (type) {
2606     case AUTH_REQUEST:
```

```
2607        draft.isAuthenticated = false;
2608        draft.isFetching = true;
2609        break;
2610      case AUTH_SUCCESS:
2611        if (els.set(AUTH_TOKEN_KEY, payload.token)) {
2612          draft.errorMessage = null;
2613          draft.isAuthenticated = true;
2614          draft.isFetching = false;
2615          draft.user = payload.user;
2616        }
2617        break;
2618      case AUTH_FAILURE:
2619        draft.errorMessage = payload.message;
2620        draft.isAuthenticated = false;
2621        draft.isFetching = false;
2622        break;
2623      case CHECK_AUTH_FAILURE:
2624        draft.isAuthenticated = false;
2625        draft.isFetching = false;
2626        break;
2627      case CHECK_AUTH_REQUEST:
2628        draft.isFetching = true;
2629        break;
2630      case CHECK_AUTH_SUCCESS:
2631        draft.errorMessage = null;
2632        draft.isFetching = false;
2633        draft.user = payload;
2634      // no default
2635    }
2636  }, state);
2637
2638  export default reducer;
2639
2640  /**
2641   * NOME DO ARQUIVO
2642   * ./reducers/index.js
2643   */
2644  import { combineReducers } from 'redux';
2645
2646  import auth, { STATE_KEY as AUTH_STATE_KEY } from './auth';
2647  import rtc, { STATE_KEY as RTC_STATE_KEY } from './rtc';
2648  import entities, { STATE_KEY as ENTITIES_STATE_KEY } from './
       entities';
2649  import socket, { STATE_KEY as SOCKET_STATE_KEY } from './socket';
2650
2651  const rootReducer = combineReducers({
2652    [AUTH_STATE_KEY]: auth,
```

```
2653   [ENTITIES_STATE_KEY]: entities,
2654   [RTC_STATE_KEY]: rtc,
2655   [SOCKET_STATE_KEY]: socket,
2656 });
2657
2658 export default rootReducer;
2659
2660 /**
2661  * NOME DO ARQUIVO
2662  * ./reducers/socket.js
2663  */
2664 import produce from 'immer';
2665
2666 import { types } from 'src/actions/socket';
2667
2668 const { SOCKET_OPEN_SUCCESS, SOCKET_CONN_ERROR,
        SOCKET_CLOSE_SUCCESS } = types;
2669
2670 export const STATE_KEY = 'socket';
2671
2672 export const state = {
2673   connected: false,
2674   error: null,
2675   sid: null,
2676 };
2677
2678 const reducer = produce((draft, { type, payload }) => {
2679   switch (type) {
2680     case SOCKET_OPEN_SUCCESS:
2681       draft.connected = true;
2682       draft.sid = payload.sid;
2683       return;
2684     case SOCKET_CONN_ERROR:
2685       draft.connected = false;
2686       draft.error = payload.error;
2687       draft.sid = null;
2688       return;
2689     case SOCKET_CLOSE_SUCCESS:
2690       draft.connected = false;
2691       draft.sid = null;
2692     // no default
2693   }
2694 }, state);
2695
2696 export default reducer;
2697
2698 /**
```

```
2699  * NOME DO ARQUIVO
2700  * ./reducers/customers.js
2701  */
2702 import produce from 'immer';
2703 import { types as serviceCallTypes } from 'src/actions/service-
        call';
2704
2705 export const STATE_KEY = 'customers';
2706
2707 export const initialState = {
2708   allIds: [],
2709   byId: {},
2710 };
2711
2712 export default produce((draft, { type, payload }) => {
2713   switch (type) {
2714     case serviceCallTypes.ENTITIES_READ_SUCCESS: {
2715       const { entities } = payload;
2716       draft.byId = { ...entities.customers };
2717       break;
2718     }
2719     // no default
2720   }
2721 }, initialState);
2722
2723 /**
2724  * NOME DO ARQUIVO
2725  * ./reducers/entities.js
2726  */
2727 import { combineReducers } from 'redux';
2728 import customers, { STATE_KEY as CUSTOMERS_STATE_KEY } from './
        customers';
2729 import serviceCalls, {
2730   STATE_KEY as SERVICE_CALLS_STATE_KEY,
2731 } from './service-call';
2732
2733 export const STATE_KEY = 'entities';
2734
2735 const reducer = combineReducers({
2736   [CUSTOMERS_STATE_KEY]: customers,
2737   [SERVICE_CALLS_STATE_KEY]: serviceCalls,
2738 });
2739
2740 export default reducer;
2741
2742 /**
2743  * NOME DO ARQUIVO
```

```
 * ./reducers/service-call.js
 */
import produce from 'immer';
import { types } from 'src/actions/service-call';

const {
  ENTITY_CREATE,
  ENTITY_DELETE,
  ENTITY_UPDATE,
  ENTITIES_READ_SUCCESS,
} = types;

export const STATE_KEY = 'serviceCalls';

export const initialState = {
  allIds: [],
  byId: {},
};

export default produce((draft, { type, payload }) => {
  switch (type) {
    case ENTITY_CREATE: {
      const { entity } = payload;
      const { id: entityId } = entity;

      draft.allIds.unshift(entityId);
      draft.byId[entityId] = entity;
      break;
    }
    case ENTITY_DELETE: {
      const { entity } = payload;
      const { id: entityId } = entity;

      draft.allIds.splice(
        draft.allIds.findIndex(svcId => svcId === entityId),
        1
      );
      delete draft.byId[entityId];
      break;
    }
    case ENTITY_UPDATE: {
      const { entity } = payload;
      const { id: entityId } = entity;
      Object.entries(entity).forEach(([key, value]) => {
        draft.byId[entityId][key] = value;
      });
      break;
```

```
2791      }
2792      case ENTITIES_READ_SUCCESS: {
2793        const { entities, result } = payload;
2794        draft.allIds = result;
2795        draft.byId = { ...entities.serviceCalls };
2796        break;
2797      }
2798      // no default
2799    }
2800  }, initialState);
2801
2802  /**
2803   * NOME DO ARQUIVO
2804   * ./reducers/rtc.js
2805   */
2806  import immerReducer from 'src/utils/immer-reducer';
2807  import { types } from 'src/actions/rtc';
2808
2809  const {
2810    PEER_CONNECT,
2811    PEER_DISCONNECT,
2812    PEER_SET,
2813    STREAM_SET,
2814    SIGNAL_RECEIVE,
2815    SIGNAL_SEND,
2816  } = types;
2817
2818  export const STATE_KEY = 'rtc';
2819
2820  export const CONNECTION_STATE = {
2821    REQUESTED: 'REQUESTED',
2822    DISCONNECTED: 'DISCONNECTED',
2823    SIGNAL_SENT: 'SIGNAL_SENT',
2824    SIGNAL_RECEIVED: 'SIGNAL_RECEIVED',
2825    CONNECTED: 'CONNECTED',
2826    CONNECTED_STREAM: 'CONNECTED_STREAM',
2827  };
2828
2829  export const initialState = {
2830    connectionState: CONNECTION_STATE.DISCONNECTED,
2831    error: null,
2832    peer: null,
2833    localStream: null,
2834    remoteStream: null,
2835  };
2836
2837  const reducer = immerReducer(
```

```
2838   {
2839     [PEER_CONNECT]: (state, payload) => {
2840       state.localStream = payload.stream;
2841       state.connectionState = CONNECTION_STATE.REQUESTED;
2842     },
2843     [PEER_DISCONNECT]: state => {
2844       state.localStream = null;
2845       state.remoteStream = null;
2846       state.peer = null;
2847       state.connectionState = CONNECTION_STATE.DISCONNECTED;
2848     },
2849     [PEER_SET]: (state, payload) => {
2850       state.peer = payload.peer;
2851       state.connectionState = CONNECTION_STATE.CONNECTED;
2852     },
2853     [SIGNAL_SEND]: state => {
2854       state.connectionState = CONNECTION_STATE.SIGNAL_SENT;
2855     },
2856     [SIGNAL_RECEIVE]: state => {
2857       state.connectionState = CONNECTION_STATE.SIGNAL_RECEIVED;
2858     },
2859     [STREAM_SET]: (state, payload) => {
2860       state.remoteStream = payload.stream;
2861       state.connectionState = CONNECTION_STATE.CONNECTED_STREAM;
2862     },
2863   },
2864   initialState
2865 );
2866
2867 export default reducer;
2868
2869 /**
2870  * NOME DO ARQUIVO
2871  * ./hocs/with-data-loading.js
2872  */
2873 import React, { Component } from 'react';
2874
2875 const withDataLoading = (action, shouldLoad) => WrappedComponent
       =>
2876   class WithDataLoading extends Component {
2877     state = {
2878       loading: true,
2879       error: null,
2880     };
2881
2882     async componentWillMount() {
2883       await this.loadData();
```

```
2884      }
2885
2886      componentWillReceiveProps(nextProps) {
2887        if (shouldLoad && shouldLoad(this.props, nextProps)) {
2888          this.setState(
2889            {
2890              loading: true,
2891            },
2892            this.loadData
2893          );
2894        }
2895      }
2896
2897      loadData = async () => {
2898        try {
2899          await action(this.props);
2900          this.setState({
2901            loading: false,
2902          });
2903        } catch (e) {
2904          this.setState({
2905            error: e,
2906            loading: false,
2907          });
2908        }
2909      };
2910
2911      render() {
2912        const { loading } = this.state;
2913        if (loading) {
2914          return <div />;
2915        }
2916
2917        return <WrappedComponent {...this.props} {...this.state}
                   />;
2918      }
2919    };
2920
2921  export default withDataLoading;
2922
2923  /**
2924   * NOME DO ARQUIVO
2925   * ./selectors/auth.js
2926   */
2927  export const isAuthenticatedSelector = state => state.auth.
       isAuthenticated; // eslint-disable-line
2928
```

```
2929 /**
2930  * NOME DO ARQUIVO
2931  * ./selectors/service-call.js
2932  */
2933 import { createSelector } from 'reselect';
2934
2935 export const serviceCallsAllIdsSelector = state =>
2936   state.entities.serviceCalls.allIds;
2937 export const serviceCallsByIdSelector = state =>
2938   state.entities.serviceCalls.byId;
2939
2940 export const serviceCallsSelector = createSelector(
2941   [serviceCallsAllIdsSelector, serviceCallsByIdSelector],
2942   (allIds, byId) => allIds.map(id => byId[id])
2943 );
2944
2945 export const endedServiceCallsSelector = createSelector(
2946   serviceCallsSelector,
2947   serviceCalls => serviceCalls.filter(serviceCall => !!
         serviceCall.endedAt)
2948 );
2949
2950 export const ratedServiceCallsSelector = createSelector(
2951   serviceCallsSelector,
2952   serviceCalls =>
2953     serviceCalls.filter(
2954       serviceCall => !!serviceCall.callRating && !!serviceCall.
             serviceRating
2955     )
2956 );
2957
2958 export const avgCallRatingSelector = createSelector(
2959   ratedServiceCallsSelector,
2960   serviceCalls =>
2961     !serviceCalls.length
2962       ? 0
2963       : serviceCalls.reduce(
2964           (sum, serviceCall) => sum + serviceCall.callRating,
2965           0
2966         ) / serviceCalls.length
2967 );
2968
2969 export const avgServiceRatingSelector = createSelector(
2970   ratedServiceCallsSelector,
2971   serviceCalls =>
2972     !serviceCalls.length
2973       ? 0
```

```
2974        : serviceCalls.reduce(
2975            (sum, serviceCall) => sum + serviceCall.serviceRating,
2976            0
2977          ) / serviceCalls.length
2978  );
2979
2980  export const solvedServiceCallsSelector = createSelector(
2981    ratedServiceCallsSelector,
2982    serviceCalls => serviceCalls.filter(serviceCall => serviceCall.
          isSolved)
2983  );
2984
2985  export const notSolvedServiceCallsSelector = createSelector(
2986    ratedServiceCallsSelector,
2987    serviceCalls => serviceCalls.filter(serviceCall => !serviceCall
          .isSolved)
2988  );
2989
2990  export const solvedStatisticsSelector = createSelector(
2991    [
2992      solvedServiceCallsSelector,
2993      notSolvedServiceCallsSelector,
2994      endedServiceCallsSelector,
2995    ],
2996    (solved, notSolved, ended) => ({
2997      solvedCalls: solved.length,
2998      notSolvedCalls: notSolved.length,
2999      notAnsweredCalls: ended.length - solved.length + notSolved.
             length,
3000    })
3001  );
3002
3003  export const serviceCallSelector = (state, id) =>
3004    state.entities.serviceCalls.byId[id];
3005
3006  export const serviceCallCustomerSelector = (state, customerId) =>
3007    state.entities.customers.byId[customerId];
3008
3009  /**
3010   * NOME DO ARQUIVO
3011   * ./store/configureStore.js
3012   */
3013  import { applyMiddleware, compose, createStore } from 'redux';
3014  import { apiMiddleware } from 'redux-api-middleware';
3015  import logger from 'redux-logger';
3016  import thunk from 'redux-thunk';
```

```
3017 import { devToolsEnhancer } from 'redux-devtools-extension'; //
       eslint-disable-line
3018 import io from 'socket.io-client';
3019
3020 import authApiMiddleware from 'src/utils/auth-api-middleware';
3021 import createSocketMiddleware from 'src/utils/socket-middleware';
3022 import rtcMiddleware from 'src/utils/rtc-middleware';
3023 import rootReducer from 'src/reducers';
3024
3025 const eventsRoot = process.env.REACT_APP_EVENTS_ROOT || '';
3026
3027 const configureStore = preloadedState => {
3028   const socket = io('/manager', {
3029     path: eventsRoot,
3030     transports: ['websocket'],
3031     autoConnect: false,
3032   });
3033   const socketMiddleware = createSocketMiddleware(socket);
3034   const middlewares = [
3035     thunk.withExtraArgument({ socket }),
3036     authApiMiddleware,
3037     apiMiddleware,
3038     socketMiddleware,
3039     rtcMiddleware,
3040     logger,
3041   ];
3042   const middlewareEnhancer = applyMiddleware(...middlewares);
3043
3044   const enhancers = [middlewareEnhancer, devToolsEnhancer()];
3045   const composedEnhancers = compose(...enhancers);
3046
3047   const store = createStore(rootReducer, preloadedState,
         composedEnhancers);
3048
3049   if (process.env.NODE_ENV !== 'production' && module.hot) {
3050     module.hot.accept('../reducers', () => store.replaceReducer(
           rootReducer));
3051   }
3052
3053   return store;
3054 };
3055
3056 export default configureStore;
```

Listing 7.2 – Projeto SAC Manager

7.3 SAC CALLER

```
1
2  /**
3   * NOME DO ARQUIVO
4   * ./constants.js
5   */
6  export const AUTH_TOKEN_KEY = 'jwt_token'; // eslint-disable-line
7  export const FSA = '@@socket/FSA';
8  export const RSSA = '@@socket/RSSA';
9
10 /**
11  * NOME DO ARQUIVO
12  * ./setupProxy.js
13  */
14 const proxy = require('http-proxy-middleware'); // eslint-disable
       -line
15
16 module.exports = function setupProxy(app) {
17   app.use(proxy('/api', { target: 'http://localhost:6000/' }));
18   app.use(proxy('/events', { target: 'ws://localhost:6000/', ws:
         true }));
19 };
20
21 /**
22  * NOME DO ARQUIVO
23  * ./index.js
24  */
25 import React from 'react';
26 import { render } from 'react-dom';
27 import Root from './Root';
28
29 render(<Root />, document.getElementById('root'));
30
31 /**
32  * NOME DO ARQUIVO
33  * ./utils/socket-middleware.js
34  */
35 import { types } from 'src/actions/socket';
36 import { FSA, RSSA } from 'src/constants';
37
38 const {
39   SOCKET_OPEN_REQUEST,
40   SOCKET_OPEN_SUCCESS,
41   SOCKET_CLOSE_REQUEST,
```

```
42    SOCKET_CLOSE_SUCCESS,
43    SOCKET_CONN_ERROR,
44  } = types;
45
46  const createSocketMiddleware = socket => ({ dispatch }) => {
47    socket.on(FSA, dispatch);
48
49    socket.on('connect', () =>
50      dispatch({
51        type: SOCKET_OPEN_SUCCESS,
52        payload: { sid: socket.id },
53      })
54    );
55    socket.on('disconnect', () => dispatch({ type:
        SOCKET_CLOSE_SUCCESS }));
56    socket.on('error', error =>
57      dispatch({ type: SOCKET_CONN_ERROR, payload: { error } })
58    );
59
60    return next => action => {
61      const socketAction = action[RSSA];
62
63      if (socketAction) {
64        const { ack, event, message, optimistic } = socketAction;
65        message.meta = {
66          ...message.meta,
67          sid: socket.id,
68        };
69        socket.emit(event || FSA, message, ack);
70        if (optimistic) {
71          dispatch(message);
72        }
73        return;
74      }
75
76      switch (action.type) {
77        case SOCKET_OPEN_REQUEST:
78          socket.connect();
79          break;
80        case SOCKET_CLOSE_REQUEST:
81          socket.disconnect();
82          break;
83        case SOCKET_CONN_ERROR:
84          // socket.connect();
85          break;
86        default:
87          break;
```

```
88      }
89
90      next(action);
91    };
92  };
93
94  export default createSocketMiddleware;
95
96  /**
97   * NOME DO ARQUIVO
98   * ./utils/immer-reducer.js
99   */
100 import produce from 'immer';
101
102 const immerReducer = (actionsMap, defaultState) => (
103   state = defaultState,
104   { type, payload }
105 ) =>
106   produce(state, draft => {
107     const action = actionsMap[type];
108     action && action(draft, payload); // eslint-disable-line
109   });
110
111 export default immerReducer;
112
113 /**
114  * NOME DO ARQUIVO
115  * ./utils/rtc-middleware.js
116  */
117 import Peer from 'simple-peer';
118
119 import {
120   disconnectPeer,
121   setPeer,
122   setStream,
123   sendSignal,
124   types,
125 } from 'src/actions/rtc';
126
127 const { PEER_CONNECT, PEER_DISCONNECT, SIGNAL_RECEIVE } = types;
128
129 const rtcMiddleware = ({ dispatch }) => {
130   let peer;
131
132   return next => action => {
133     const { meta, payload, type } = action;
134     switch (type) {
```

```
135       case PEER_CONNECT: {
136         const { stream } = payload;
137         const { room } = meta;
138
139         peer = new Peer({ initiator: false, stream });
140         peer.on('signal', signal => {
141           dispatch(sendSignal({ room, signal }));
142         });
143
144         peer.on('connect', () => dispatch(setPeer(peer)));
145         peer.on('stream', peerStream => dispatch(setStream(
              peerStream)));
146
147         peer.on('close', () => next(disconnectPeer()));
148         peer.on('error', err => dispatch(disconnectPeer({ err }))
              );
149
150         break;
151       }
152       case PEER_DISCONNECT: {
153         peer.destroy();
154         return;
155       }
156       case SIGNAL_RECEIVE: {
157         const { signal } = payload;
158         peer.signal(signal);
159         break;
160       }
161       default:
162         break;
163     }
164
165     next(action);
166   };
167 };
168
169 export default rtcMiddleware;
170
171 /**
172  * NOME DO ARQUIVO
173  * ./utils/create-constants.js
174  */
175 export default (prefix = '') => (...keys) =>
176   keys.reduce((obj, key) => ({ ...obj, [key]: prefix + key }),
        {});
177
178 /**
```

```
179  * NOME DO ARQUIVO
180  * ./styles/Global.js
181  */
182 /* stylelint-disable */
183 import { createGlobalStyle } from 'styled-components';
184
185 const Global = createGlobalStyle`
186   /*! normalize.css v8.0.0 | MIT License | github.com/necolas/
        normalize.css */
187
188   /* Document
189     ========================================================================
          */
190
191   /**
192   * 1. Correct the line height in all browsers.
193   * 2. Prevent adjustments of font size after orientation changes
        in iOS.
194   */
195
196   html {
197     line-height: 1.15; /* 1 */
198     -webkit-text-size-adjust: 100%; /* 2 */
199   }
200
201   /* Sections
202     ========================================================================
          */
203
204   /**
205   * Remove the margin in all browsers.
206   */
207
208   body {
209     margin: 0;
210   }
211
212   /**
213   * Correct the font size and margin on h1 elements within
        section and
214   * article contexts in Chrome, Firefox, and Safari.
215   */
216
217   h1 {
218     font-size: 2em;
219     margin: 0.67em 0;
220   }
```

```
221
222   /* Grouping content
223     ========================================================================
          */
224
225   /**
226   * 1. Add the correct box sizing in Firefox.
227   * 2. Show the overflow in Edge and IE.
228   */
229
230   hr {
231     box-sizing: content-box; /* 1 */
232     height: 0; /* 1 */
233     overflow: visible; /* 2 */
234   }
235
236   /**
237   * 1. Correct the inheritance and scaling of font size in all
        browsers.
238   * 2. Correct the odd em font sizing in all browsers.
239   */
240
241   pre {
242     font-family: monospace, monospace; /* 1 */
243     font-size: 1em; /* 2 */
244   }
245
246   /* Text-level semantics
247     ========================================================================
          */
248
249   /**
250   * Remove the gray background on active links in IE 10.
251   */
252
253   a {
254     background-color: transparent;
255   }
256
257   /**
258   * 1. Remove the bottom border in Chrome 57-
259   * 2. Add the correct text decoration in Chrome, Edge, IE, Opera
        , and Safari.
260   */
261
262   abbr[title] {
263     border-bottom: none; /* 1 */
```

```
264     text-decoration: underline; /* 2 */
265     text-decoration: underline dotted; /* 2 */
266   }
267
268   /**
269   * Add the correct font weight in Chrome, Edge, and Safari.
270   */
271
272   b,
273   strong {
274     font-weight: bolder;
275   }
276
277   /**
278   * 1. Correct the inheritance and scaling of font size in all
        browsers.
279   * 2. Correct the odd em font sizing in all browsers.
280   */
281
282   code,
283   kbd,
284   samp {
285     font-family: monospace, monospace; /* 1 */
286     font-size: 1em; /* 2 */
287   }
288
289   /**
290   * Add the correct font size in all browsers.
291   */
292
293   small {
294     font-size: 80%;
295   }
296
297   /**
298   * Prevent sub and sup elements from affecting the line height
        in
299   * all browsers.
300   */
301
302   sub,
303   sup {
304     font-size: 75%;
305     line-height: 0;
306     position: relative;
307     vertical-align: baseline;
308   }
```

```
309
310   sub {
311     bottom: -0.25em;
312   }
313
314   sup {
315     top: -0.5em;
316   }
317
318   /* Embedded content
319     ========================================================================
          */
320
321   /**
322   * Remove the border on images inside links in IE 10.
323   */
324
325   img {
326     border-style: none;
327   }
328
329   /* Forms
330     ========================================================================
          */
331
332   /**
333   * 1. Change the font styles in all browsers.
334   * 2. Remove the margin in Firefox and Safari.
335   */
336
337   button,
338   input,
339   optgroup,
340   select,
341   textarea {
342     font-family: inherit; /* 1 */
343     font-size: 100%; /* 1 */
344     line-height: 1.15; /* 1 */
345     margin: 0; /* 2 */
346   }
347
348   /**
349   * Show the overflow in IE.
350   * 1. Show the overflow in Edge.
351   */
352
353   button,
```

```
354  input { /* 1 */
355    overflow: visible;
356  }
357
358  /**
359  * Remove the inheritance of text transform in Edge, Firefox,
       and IE.
360  * 1. Remove the inheritance of text transform in Firefox.
361  */
362
363  button,
364  select { /* 1 */
365    text-transform: none;
366  }
367
368  /**
369  * Correct the inability to style clickable types in iOS and
       Safari.
370  */
371
372  button,
373  [type="button"],
374  [type="reset"],
375  [type="submit"] {
376    -webkit-appearance: button;
377  }
378
379  /**
380  * Remove the inner border and padding in Firefox.
381  */
382
383  button::-moz-focus-inner,
384  [type="button"]::-moz-focus-inner,
385  [type="reset"]::-moz-focus-inner,
386  [type="submit"]::-moz-focus-inner {
387    border-style: none;
388    padding: 0;
389  }
390
391  /**
392  * Restore the focus styles unset by the previous rule.
393  */
394
395  button:-moz-focusring,
396  [type="button"]:-moz-focusring,
397  [type="reset"]:-moz-focusring,
398  [type="submit"]:-moz-focusring {
```

```
399     outline: 1px dotted ButtonText;
400   }
401
402   /**
403   * Correct the padding in Firefox.
404   */
405
406   fieldset {
407     padding: 0.35em 0.75em 0.625em;
408   }
409
410   /**
411   * 1. Correct the text wrapping in Edge and IE.
412   * 2. Correct the color inheritance from fieldset elements in IE
        .
413   * 3. Remove the padding so developers are not caught out when
        they zero out
414   *    fieldset elements in all browsers.
415   */
416
417   legend {
418     box-sizing: border-box; /* 1 */
419     color: inherit; /* 2 */
420     display: table; /* 1 */
421     max-width: 100%; /* 1 */
422     padding: 0; /* 3 */
423     white-space: normal; /* 1 */
424   }
425
426   /**
427   * Add the correct vertical alignment in Chrome, Firefox, and
        Opera.
428   */
429
430   progress {
431     vertical-align: baseline;
432   }
433
434   /**
435   * Remove the default vertical scrollbar in IE 10+.
436   */
437
438   textarea {
439     overflow: auto;
440   }
441
442   /**
```

```
443   * 1. Add the correct box sizing in IE 10.
444   * 2. Remove the padding in IE 10.
445   */
446
447   [type="checkbox"],
448   [type="radio"] {
449     box-sizing: border-box; /* 1 */
450     padding: 0; /* 2 */
451   }
452
453   /**
454   * Correct the cursor style of increment and decrement buttons
          in Chrome.
455   */
456
457   [type="number"]::-webkit-inner-spin-button,
458   [type="number"]::-webkit-outer-spin-button {
459     height: auto;
460   }
461
462   /**
463   * 1. Correct the odd appearance in Chrome and Safari.
464   * 2. Correct the outline style in Safari.
465   */
466
467   [type="search"] {
468     -webkit-appearance: textfield; /* 1 */
469     outline-offset: -2px; /* 2 */
470   }
471
472   /**
473   * Remove the inner padding in Chrome and Safari on macOS.
474   */
475
476   [type="search"]::-webkit-search-decoration {
477     -webkit-appearance: none;
478   }
479
480   /**
481   * 1. Correct the inability to style clickable types in iOS and
          Safari.
482   * 2. Change font properties to inherit in Safari.
483   */
484
485   ::-webkit-file-upload-button {
486     -webkit-appearance: button; /* 1 */
487     font: inherit; /* 2 */
```

```
488   }
489
490   /* Interactive
491     ========================================================================
          */
492
493   /*
494   * Add the correct display in Edge, IE 10+, and Firefox.
495   */
496
497   details {
498     display: block;
499   }
500
501   /*
502   * Add the correct display in all browsers.
503   */
504
505   summary {
506     display: list-item;
507   }
508
509   /* Misc
510     ========================================================================
          */
511
512   /**
513   * Add the correct display in IE 10+.
514   */
515
516   template {
517     display: none;
518   }
519
520   /**
521   * Add the correct display in IE 10.
522   */
523
524   [hidden] {
525     display: none;
526   }
527
528   /******************************
529               Page
530   ******************************/
531
532   html,
```

```
533 body {
534   height: 100%;
535   width: 100%;
536 }
537
538 html {
539   font-size: 14px;
540 }
541
542 body {
543   margin: 0px;
544   padding: 0px;
545   overflow-x: hidden;
546   min-width: 320px;
547   background: #FFFFFF;
548   font-family: 'Lato', 'Helvetica Neue', Arial, Helvetica, sans
        -serif;
549   font-size: 14px;
550   line-height: 1.4285em;
551   color: rgba(0, 0, 0, 0.87);
552   font-smoothing: antialiased;
553 }
554
555 /*******************************
556             Headers
557 *******************************/
558
559 h1,
560 h2,
561 h3,
562 h4,
563 h5 {
564   font-family: 'Lato', 'Helvetica Neue', Arial, Helvetica, sans
        -serif;
565   line-height: 1.28571429em;
566   margin: calc(2rem - 0.14285714em) 0em 1rem;
567   font-weight: bold;
568   padding: 0em;
569 }
570
571 h1 {
572   min-height: 1rem;
573   font-size: 2rem;
574 }
575
576 h2 {
577   font-size: 1.71428571rem;
```

```
578   }
579
580   h3 {
581     font-size: 1.28571429rem;
582   }
583
584   h4 {
585     font-size: 1.07142857rem;
586   }
587
588   h5 {
589     font-size: 1rem;
590   }
591
592   h1:first-child,
593   h2:first-child,
594   h3:first-child,
595   h4:first-child,
596   h5:first-child {
597     margin-top: 0em;
598   }
599
600   h1:last-child,
601   h2:last-child,
602   h3:last-child,
603   h4:last-child,
604   h5:last-child {
605     margin-bottom: 0em;
606   }
607
608   /*******************************
609                 Text
610   *******************************/
611
612   p {
613     margin: 0em 0em 1em;
614     line-height: 1.4285em;
615   }
616
617   p:first-child {
618     margin-top: 0em;
619   }
620
621   p:last-child {
622     margin-bottom: 0em;
623   }
624
```

```
625 /*--------------------
626         Links
627 ---------------------*/
628
629 a {
630   color: #4183C4;
631   text-decoration: none;
632 }
633
634 a:hover {
635   color: #1e70bf;
636   text-decoration: none;
637 }
638
639 /*******************************
640         Scrollbars
641 *******************************/
642
643 /*******************************
644           Highlighting
645 *******************************/
646
647 /* Site */
648
649 ::-webkit-selection {
650   background-color: #CCE2FF;
651   color: rgba(0, 0, 0, 0.87);
652 }
653
654 ::-moz-selection {
655   background-color: #CCE2FF;
656   color: rgba(0, 0, 0, 0.87);
657 }
658
659 ::selection {
660   background-color: #CCE2FF;
661   color: rgba(0, 0, 0, 0.87);
662 }
663
664 /* Form */
665
666 textarea::-webkit-selection,
667 input::-webkit-selection {
668   background-color: rgba(100, 100, 100, 0.4);
669   color: rgba(0, 0, 0, 0.87);
670 }
671
```

```
672   textarea::-moz-selection,
673   input::-moz-selection {
674     background-color: rgba(100, 100, 100, 0.4);
675     color: rgba(0, 0, 0, 0.87);
676   }
677
678   textarea::selection,
679   input::selection {
680     background-color: rgba(100, 100, 100, 0.4);
681     color: rgba(0, 0, 0, 0.87);
682   }
683
684   /* Force Simple Scrollbars */
685
686   body ::-webkit-scrollbar {
687     -webkit-appearance: none;
688     width: 10px;
689     height: 10px;
690   }
691
692   body ::-webkit-scrollbar-track {
693     background: rgba(0, 0, 0, 0.1);
694     border-radius: 0px;
695   }
696
697   body ::-webkit-scrollbar-thumb {
698     cursor: pointer;
699     border-radius: 5px;
700     background: rgba(0, 0, 0, 0.25);
701     -webkit-transition: color 0.2s ease;
702     transition: color 0.2s ease;
703   }
704
705   body ::-webkit-scrollbar-thumb:window-inactive {
706     background: rgba(0, 0, 0, 0.15);
707   }
708
709   body ::-webkit-scrollbar-thumb:hover {
710     background: rgba(128, 135, 139, 0.8);
711   }
712
713   #root, #app {
714     height: 100%;
715     width: 100%;
716   }
717 `;
718
```

```
719 export default Global;
720
721 /**
722  * NOME DO ARQUIVO
723  * ./styles/__tests__/Button.js
724  */
725 import React from 'react';
726 import { render } from 'react-testing-library';
727
728 import Button from '../Button';
729
730 describe('[styled-component] Button', () => {
731   it('matches snapshot', () => {
732     const { container } = render(<Button />);
733     expect(container.firstChild).toMatchSnapshot();
734   });
735 });
736
737 /**
738  * NOME DO ARQUIVO
739  * ./styles/Button.js
740  */
741 import styled from 'styled-components';
742
743 const Button = styled.button`
744   font-size: 1em;
745   margin: 1em;
746   padding: 0.25em 1em;
747   border-radius: 3px;
748   color: ${props => props.theme.main};
749   border: 2px solid ${props => props.theme.main};
750 `;
751
752 export default Button;
753
754 /**
755  * NOME DO ARQUIVO
756  * ./styles/FullScreenContainer.js
757  */
758 import styled from 'styled-components';
759 import { Flex } from '@rebass/grid';
760
761 const FullScreenContainer = styled(Flex)`
762   min-height: 100vh;
763   min-width: 100vw;
764 `;
765
```

```
766 export default FullScreenContainer;
767
768 /**
769  * NOME DO ARQUIVO
770  * ./Root.js
771  */
772 import React from 'react';
773 import { BrowserRouter as Router, Route, Switch } from 'react-
      router-dom';
774 import { Provider } from 'react-redux';
775 import { grommet } from 'grommet/themes';
776 import { Grommet } from 'grommet';
777
778 import AppRoute from 'src/routes/AppRoute';
779 import { APP_PATH } from 'src/routes/paths';
780 import configureStore from 'src/store/configureStore';
781 import Global from 'src/styles/Global';
782
783 const store = configureStore();
784
785 const Root = () => (
786   <Provider store={store}>
787     <Grommet theme={grommet}>
788       <Router>
789         <>
790           <Global />
791           <Switch>
792             <Route path={APP_PATH} component={AppRoute} />
793           </Switch>
794         </>
795       </Router>
796     </Grommet>
797   </Provider>
798 );
799
800 export default Root;
801
802 /**
803  * NOME DO ARQUIVO
804  * ./actions/socket.js
805  */
806 import createConstants from 'src/utils/create-constants';
807
808 export const types = createConstants('@socket/')(
809   'SOCKET_OPEN_REQUEST',
810   'SOCKET_OPEN_SUCCESS',
811   'SOCKET_CLOSE_REQUEST',
```

```
812   'SOCKET_CLOSE_SUCCESS',
813   'SOCKET_CONN_ERROR',
814   'FSA',
815   'RSSA'
816 );
817
818 const { SOCKET_OPEN_REQUEST } = types;
819
820 export const connectSocket = () => ({
821   type: SOCKET_OPEN_REQUEST,
822 });
823
824 /**
825  * NOME DO ARQUIVO
826  * ./actions/service-call.js
827  */
828 import { RSSA } from 'src/constants';
829 import createConstants from 'src/utils/create-constants';
830
831 export const types = createConstants('@@service-call/')(
832   'ENTITY_CREATE',
833   'ENTITY_DELETE',
834   'ENTITY_UPDATE',
835   'SERVICE_CALL_SET_ACTIVE'
836 );
837
838 const {
839   ENTITY_CREATE,
840   ENTITY_DELETE,
841   ENTITY_UPDATE,
842   SERVICE_CALL_SET_ACTIVE,
843 } = types;
844
845 export const createServiceCall = ({ customerId }) => dispatch =>
846   dispatch({
847     [RSSA]: {
848       event: ENTITY_CREATE,
849       message: { data: { customerId } },
850       ack: (_, entity) => {
851         dispatch({ type: ENTITY_CREATE, payload: { entity } });
852         dispatch({
853           type: SERVICE_CALL_SET_ACTIVE,
854           payload: { serviceCallId: entity.id },
855         });
856       },
857     },
858   });
```

```
859
860 export const updateServiceCall = data => ({
861   [RSSA]: {
862     event: ENTITY_UPDATE,
863     message: { data },
864   },
865 });
866
867 export const deleteServiceCall = ({ serviceCallId }) => ({
868   [RSSA]: {
869     event: ENTITY_DELETE,
870     message: { data: { serviceCallId } },
871   },
872 });
873
874 /**
875  * NOME DO ARQUIVO
876  * ./actions/rtc.js
877  */
878 import { RSSA } from 'src/constants';
879 import createConstants from 'src/utils/create-constants';
880
881 const createRtcConstants = createConstants('@@rtc/');
882
883 export const types = createRtcConstants(
884   'PEER_CONNECT',
885   'PEER_DISCONNECT',
886   'PEER_SET',
887   'STREAM_SET',
888   'SIGNAL_RECEIVE',
889   'SIGNAL_SEND'
890 );
891
892 const {
893   PEER_CONNECT,
894   PEER_DISCONNECT,
895   PEER_SET,
896   STREAM_SET,
897   SIGNAL_RECEIVE,
898   SIGNAL_SEND,
899 } = types;
900
901 export const connectPeer = ({ room, stream }) => ({
902   [RSSA]: {
903     event: PEER_CONNECT,
904     message: {
905       type: PEER_CONNECT,
```

```
906         payload: { stream },
907         meta: { namespace: '/manager', room },
908       },
909       optimistic: true,
910     },
911   });
912
913   export const disconnectPeer = ({ err } = {}) => ({
914     type: PEER_DISCONNECT,
915     payload: { err },
916   });
917
918   export const sendSignal = ({ room, signal }) => ({
919     [RSSA]: {
920       event: SIGNAL_SEND,
921       message: {
922         type: SIGNAL_RECEIVE,
923         payload: { signal },
924         meta: { namespace: '/manager', room },
925       },
926     },
927   });
928
929   export const receiveSignal = signal => ({
930     type: SIGNAL_RECEIVE,
931     payload: { signal },
932   });
933
934   export const setPeer = peer => ({
935     type: PEER_SET,
936     payload: { peer },
937   });
938
939   export const setStream = stream => ({
940     type: STREAM_SET,
941     payload: { stream },
942   });
943
944   /**
945    * NOME DO ARQUIVO
946    * ./containers/ServiceCallFeedbackForm/index.js
947    */
948   import React from 'react';
949   import PropTypes from 'prop-types';
950   import { Box, Heading, RadioButton, Button } from 'grommet';
951   import Rating from 'react-rating';
952   import { Star } from 'grommet-icons';
```

```
import { connect } from 'react-redux';
import { compose, withHandlers, withStateHandlers } from '
    recompose';

import * as serviceCallActions from 'src/actions/service-call';

const validate = ({ callRating, serviceRating, isSolved }) => {
  if (callRating === 0) return false;
  if (serviceRating === 0) return false;
  if (!isSolved) return false;
  return true;
};

const ServiceCallFeedbackForm = ({
  callRating,
  isSolved,
  serviceRating,
  setCallRating,
  setIsSolved,
  setServiceRating,
  onSubmit,
}) => (
  <form onSubmit={onSubmit}>
    <Heading level="4">CALL</Heading>
    <Rating
      emptySymbol={<Star />}
      fullSymbol={<Star color="neutral-5" />}
      initialRating={callRating}
      onChange={setCallRating}
      stop={5}
    />
    <Heading level="4">SERVICE</Heading>
    <Rating
      emptySymbol={<Star />}
      fullSymbol={<Star color="neutral-5" />}
      initialRating={serviceRating}
      onChange={setServiceRating}
      stop={5}
    />
    <Heading level="4">DID YOU SOLVE YOUR PROBLEM?</Heading>
    <Box direction="row" gap="medium">
      <RadioButton
        label="YES"
        name="is-solved"
        value="solved"
        checked={isSolved === 'solved'}
        onChange={setIsSolved}
```

```
999         />
1000        <RadioButton
1001          label="NO"
1002          name="is-solved"
1003          value="not-solved"
1004          checked={isSolved === 'not-solved'}
1005          onChange={setIsSolved}
1006        />
1007      </Box>
1008      <Button
1009        color="status-ok"
1010        label="Send feedback"
1011        margin={{ top: 'medium' }}
1012        type="submit"
1013        disabled={!validate({ callRating, serviceRating, isSolved
              })}
1014        primary
1015      />
1016    </form>
1017  );
1018
1019  ServiceCallFeedbackForm.propTypes = {
1020    callRating: PropTypes.number.isRequired,
1021    isSolved: PropTypes.oneOfType([PropTypes.bool, PropTypes.string
        ]).isRequired,
1022    serviceRating: PropTypes.number.isRequired,
1023    setCallRating: PropTypes.func.isRequired,
1024    setIsSolved: PropTypes.func.isRequired,
1025    setServiceRating: PropTypes.func.isRequired,
1026    onSubmit: PropTypes.func.isRequired,
1027  };
1028
1029  ServiceCallFeedbackForm.defaultProps = {};
1030
1031  export default compose(
1032    connect(
1033      null,
1034      {
1035        updateServiceCall: serviceCallActions.updateServiceCall,
1036      }
1037    ),
1038    withStateHandlers(
1039      {
1040        callRating: 0,
1041        serviceRating: 0,
1042        isSolved: false,
1043      },
```

```
1044      {
1045        setServiceRating: () => value => ({
1046          serviceRating: value,
1047        }),
1048        setCallRating: () => value => ({
1049          callRating: value,
1050        }),
1051        setIsSolved: () => event => ({
1052          isSolved: event.target.value,
1053        }),
1054      }
1055    ),
1056    withHandlers({
1057      onSubmit: ({
1058        updateServiceCall,
1059        id,
1060        serviceRating,
1061        callRating,
1062        isSolved,
1063      }) => evt => {
1064        updateServiceCall({
1065          id,
1066          serviceRating,
1067          callRating,
1068          isSolved: isSolved === 'solved',
1069        });
1070        evt.preventDefault();
1071      },
1072    })
1073  )(ServiceCallFeedbackForm);
1074
1075  /**
1076   * NOME DO ARQUIVO
1077   * ./containers/VideoChat/index.js
1078   */
1079  import { connect } from 'react-redux';
1080  import { compose, lifecycle, withHandlers } from 'recompose';
1081
1082  import * as rtcActions from 'src/actions/rtc';
1083
1084  import VideoChat from './components/VideoChat';
1085
1086  const mapStateToProps = state => ({
1087    localStream: state.rtc.localStream,
1088    remoteStream: state.rtc.remoteStream,
1089  });
1090
```

```
1091 const mapDispatchToProps = {
1092   connectPeer: rtcActions.connectPeer,
1093   disconnectPeer: rtcActions.disconnectPeer,
1094 };
1095
1096 const constraints = {
1097   video: {
1098     width: { max: 1920 },
1099     height: { max: 1920 },
1100   },
1101   audio: true,
1102 };
1103
1104 const handleError = error => {
1105   if (error.name === 'ConstraintNotSatisfiedError') {
1106     const v = constraints.video;
1107     console.log(
1108       `The resolution ${v.width.exact}x${
1109         v.height.exact
1110       } px is not supported by your device.`
1111     );
1112   } else if (error.name === 'PermissionDeniedError') {
1113     console.log(
1114       'Permissions have not been granted to use your camera and ' +
1115         'microphone, you need to allow the page access to your devices in ' +
1116         'order for the demo to work.'
1117     );
1118   }
1119   console.log(`getUserMedia error: ${error.name}`, error);
1120 };
1121
1122 export default compose(
1123   connect(
1124     mapStateToProps,
1125     mapDispatchToProps
1126   ),
1127   withHandlers({
1128     onClickHangUp: ({ disconnectPeer, localStream }) => () => {
1129       if (localStream) {
1130         localStream.getTracks().forEach(track => track.stop());
1131       }
1132       disconnectPeer();
1133     },
1134   }),
1135   lifecycle({
```

```
1136      componentDidMount() {
1137        const { room, connectPeer } = this.props;
1138
1139        navigator.mediaDevices
1140          .getUserMedia(constraints)
1141          .then(stream => connectPeer({ room, stream }))
1142          .catch(handleError);
1143      },
1144    })
1145  )(VideoChat);
1146
1147  /**
1148   * NOME DO ARQUIVO
1149   * ./containers/VideoChat/styles/VideoChatOverlay.js
1150   */
1151  import styled, { keyframes } from 'styled-components';
1152  import { Box } from 'grommet';
1153
1154  const fadeOut = keyframes`
1155    0% {
1156      opacity: 1;
1157    }
1158    100% {
1159      opacity: 0;
1160    }
1161  `;
1162
1163  const VideoChatOverlay = styled(Box).attrs({
1164    round: true,
1165  })`
1166    animation: 0.3s ${fadeOut} 3.2s ease-in forwards;
1167    background-color: rgba(221, 221, 221, 0.2);
1168    height: 100%;
1169    left: 0;
1170    overflow: hidden;
1171    position: absolute;
1172    top: 0;
1173    width: 100%;
1174
1175    &:active,
1176    &:hover {
1177      animation: unset;
1178      opacity: 1;
1179      transition: unset;
1180    }
1181  `;
1182
```

```
1183 export default VideoChatOverlay;
1184
1185 /**
1186  * NOME DO ARQUIVO
1187  * ./containers/VideoChat/styles/VideoBox.js
1188  */
1189 import styled from 'styled-components';
1190 import { Box } from 'grommet';
1191
1192 const VideoBox = styled(Box)`
1193   position: absolute;
1194   top: 12px;
1195   right: 12px;
1196   width: 50%;
1197
1198   @media screen and (min-width: 36em) {
1199     width: 25%;
1200   }
1201 `;
1202
1203 export default VideoBox;
1204
1205 /**
1206  * NOME DO ARQUIVO
1207  * ./containers/VideoChat/components/VideoChat.js
1208  */
1209 import React, { Component, createRef } from 'react';
1210 import PropTypes from 'prop-types';
1211 import { Button, Box, Layer } from 'grommet';
1212 import { Stretch } from 'styled-loaders-react';
1213
1214 import VideoBox from '../styles/VideoBox';
1215 import VideoChatOverlay from '../styles/VideoChatOverlay';
1216
1217 class VideoChat extends Component {
1218   constructor(props) {
1219     super(props);
1220
1221     this.localVideoRef = createRef();
1222     this.remoteVideoRef = createRef();
1223   }
1224
1225   componentDidUpdate() {
1226     const { localStream, remoteStream } = this.props;
1227
1228     if (localStream) {
1229       this.localVideoRef.current.srcObject = localStream;
```

```
1230      }
1231
1232      if (remoteStream) {
1233        this.remoteVideoRef.current.srcObject = remoteStream;
1234      }
1235    }
1236
1237    render() {
1238      const { localStream, remoteStream, onClickHangUp } = this.
             props;
1239      return (
1240        <Layer
1241          position="center"
1242          margin="large"
1243          style={{
1244            backgroundColor: 'transparent',
1245            height: '100%',
1246            overflow: 'hidden',
1247          }}
1248          modal
1249          full
1250        >
1251          <Box
1252            align="center"
1253            justify="center"
1254            background="#000"
1255            overflow="hidden"
1256            elevation="xlarge"
1257            round
1258            fill
1259          >
1260            {remoteStream ? (
1261              <video
1262                style={{
1263                  width: '100%',
1264                  height: '100%',
1265                  objectFit: 'contain',
1266                  transform: 'scaleX(-1)',
1267                }}
1268                ref={this.remoteVideoRef}
1269                autoPlay
1270                muted
1271              />
1272            ) : (
1273              <Stretch color="#7D4CDB" size="100px" />
1274            )}
1275            <VideoChatOverlay>
```

```
1276              <VideoBox
1277                align="center"
1278                justify="center"
1279                background="#000"
1280                elevation="xlarge"
1281                overflow="hidden"
1282                round
1283              >
1284                {localStream ? (
1285                  <video
1286                    style={{
1287                      width: '100%',
1288                      height: '100%',
1289                      objectFit: 'contain',
1290                      transform: 'scaleX(-1)',
1291                    }}
1292                    ref={this.localVideoRef}
1293                    autoPlay
1294                    muted
1295                  />
1296                ) : (
1297                  <Stretch color="#7D4CDB" size="100px" />
1298                )}
1299              </VideoBox>
1300              <Button
1301                color="status-critical"
1302                label="Hang up"
1303                style={{
1304                  position: 'absolute',
1305                  bottom: '24px',
1306                  left: '50%',
1307                  transform: 'translateX(-50%)',
1308                }}
1309                onClick={onClickHangUp}
1310                primary
1311              />
1312            </VideoChatOverlay>
1313          </Box>
1314        </Layer>
1315      );
1316    }
1317  }
1318
1319  VideoChat.propTypes = {
1320    localStream: PropTypes.shape({
1321      id: PropTypes.string,
1322    }),
```

```
1323   remoteStream: PropTypes.shape({
1324     id: PropTypes.string,
1325   }),
1326   onClickHangUp: PropTypes.func,
1327 };
1328
1329 VideoChat.defaultProps = {
1330   localStream: null,
1331   remoteStream: null,
1332   onClickHangUp: null,
1333 };
1334
1335 export default VideoChat;
1336
1337 /**
1338  * NOME DO ARQUIVO
1339  * ./setupTests.js
1340  */
1341 import 'jest-styled-components';
1342
1343 /**
1344  * NOME DO ARQUIVO
1345  * ./pages/AppPage/index.js
1346  */
1347 import React from 'react';
1348 import PropTypes from 'prop-types';
1349 import { Box, Grommet, Heading } from 'grommet';
1350
1351 import ServiceCallFeedbackForm from 'src/containers/
        ServiceCallFeedbackForm';
1352 import VideoChat from 'src/containers/VideoChat';
1353
1354 const AppPage = ({ serviceCall }) => (
1355   <Grommet full>
1356     {serviceCall &&
1357       serviceCall.endedAt && (
1358         <Box align="center" justify="center" background="light-3"
                fill>
1359           {serviceCall.callRating > 0 ? (
1360             <Heading level="1">THANKS FOR RATING!</Heading>
1361           ) : (
1362             <ServiceCallFeedbackForm id={serviceCall.id} />
1363           )}
1364         </Box>
1365       )}
1366     {serviceCall && !serviceCall.endedAt && <VideoChat room={
        serviceCall.id} />}
```

```
1367   </Grommet>
1368 );
1369
1370 AppPage.propTypes = {
1371   serviceCall: PropTypes.shape({
1372     id: PropTypes.string,
1373   }),
1374 };
1375
1376 AppPage.defaultProps = {
1377   serviceCall: null,
1378 };
1379
1380 export default AppPage;
1381
1382 /**
1383  * NOME DO ARQUIVO
1384  * ./pages/LoadingPage/index.js
1385  */
1386 import React from 'react';
1387 import { Grommet, Box } from 'grommet';
1388 import { Stretch } from 'styled-loaders-react';
1389
1390 const LoadingPage = () => (
1391   <Grommet full>
1392     <Box align="center" background="dark-1" justify="center" fill
          >
1393       <Stretch color="#7D4CDB" size="100px" />
1394     </Box>
1395   </Grommet>
1396 );
1397
1398 export default LoadingPage;
1399
1400 /**
1401  * NOME DO ARQUIVO
1402  * ./routes/paths.js
1403  */
1404 export const APP_PATH = '/:customerId'; // eslint-disable-line
1405
1406 /**
1407  * NOME DO ARQUIVO
1408  * ./routes/AppRoute/index.js
1409  */
1410 import { connect } from 'react-redux';
1411 import { branch, compose, lifecycle, renderComponent } from '
    recompose';
```

```
1412
1413 import AppPage from 'src/pages/AppPage';
1414 import LoadingPage from 'src/pages/LoadingPage';
1415 import * as serviceCallActions from 'src/actions/service-call';
1416
1417 const getActiveServiceCall = state => {
1418   const serviceCallId = state.ui.activeServiceCall;
1419   const serviceCall = state.entities.serviceCalls.byId[
          serviceCallId];
1420
1421   return serviceCall;
1422 };
1423
1424 const mapStateToProps = state => ({
1425   serviceCall: getActiveServiceCall(state),
1426 });
1427
1428 const mapDispatchToProps = {
1429   createServiceCall: serviceCallActions.createServiceCall,
1430 };
1431
1432 export default compose(
1433   connect(
1434     mapStateToProps,
1435     mapDispatchToProps
1436   ),
1437   lifecycle({
1438     componentDidMount() {
1439       const { createServiceCall, match } = this.props;
1440       const { customerId } = match.params;
1441
1442       createServiceCall({ customerId });
1443     },
1444   }),
1445   branch(({ serviceCall }) => !serviceCall, renderComponent(
          LoadingPage))
1446 )(AppPage);
1447
1448 /**
1449  * NOME DO ARQUIVO
1450  * ./reducers/ui.js
1451  */
1452 import { combineReducers } from 'redux';
1453 import { types } from 'src/actions/service-call';
1454
1455 export const STATE_KEY = 'ui';
1456
```

```
1457 const activeServiceCall = (state = null, { type, payload }) => {
1458   switch (type) {
1459     case types.SERVICE_CALL_SET_ACTIVE: {
1460       const { serviceCallId } = payload;
1461       return serviceCallId;
1462     }
1463     default:
1464       return state;
1465   }
1466 };
1467
1468 const reducer = combineReducers({
1469   activeServiceCall,
1470 });
1471
1472 export default reducer;
1473
1474 /**
1475  * NOME DO ARQUIVO
1476  * ./reducers/index.js
1477  */
1478 import { combineReducers } from 'redux';
1479
1480 import entities, { STATE_KEY as ENTITIES_STATE_KEY } from './
       entities';
1481 import rtc, { STATE_KEY as RTC_STATE_KEY } from './rtc';
1482 import socket, { STATE_KEY as SOCKET_STATE_KEY } from './socket';
1483 import ui, { STATE_KEY as UI_STATE_KEY } from './ui';
1484
1485 const rootReducer = combineReducers({
1486   [ENTITIES_STATE_KEY]: entities,
1487   [RTC_STATE_KEY]: rtc,
1488   [SOCKET_STATE_KEY]: socket,
1489   [UI_STATE_KEY]: ui,
1490 });
1491
1492 export default rootReducer;
1493
1494 /**
1495  * NOME DO ARQUIVO
1496  * ./reducers/socket.js
1497  */
1498 import produce from 'immer';
1499
1500 import { types } from 'src/actions/socket';
1501
```

```
1502 const { SOCKET_OPEN_SUCCESS, SOCKET_CONN_ERROR,
       SOCKET_CLOSE_SUCCESS } = types;
1503
1504 export const STATE_KEY = 'socket';
1505
1506 export const state = {
1507   connected: false,
1508   error: null,
1509   sid: null,
1510 };
1511
1512 const reducer = produce((draft, { type, payload }) => {
1513   switch (type) {
1514     case SOCKET_OPEN_SUCCESS:
1515       draft.connected = true;
1516       draft.sid = payload.sid;
1517       return;
1518     case SOCKET_CONN_ERROR:
1519       draft.connected = false;
1520       draft.error = payload.error;
1521       draft.sid = null;
1522       return;
1523     case SOCKET_CLOSE_SUCCESS:
1524       draft.connected = false;
1525       draft.sid = null;
1526     // no default
1527   }
1528 }, state);
1529
1530 export default reducer;
1531
1532 /**
1533  * NOME DO ARQUIVO
1534  * ./reducers/entities.js
1535  */
1536 import { combineReducers } from 'redux';
1537 import serviceCalls, {
1538   STATE_KEY as SERVICE_CALLS_STATE_KEY,
1539 } from './service-call';
1540
1541 export const STATE_KEY = 'entities';
1542
1543 const reducer = combineReducers({
1544   [SERVICE_CALLS_STATE_KEY]: serviceCalls,
1545 });
1546
1547 export default reducer;
```

```
1548
1549 /**
1550  * NOME DO ARQUIVO
1551  * ./reducers/service-call.js
1552  */
1553 import produce from 'immer';
1554 import { types } from 'src/actions/service-call';
1555
1556 const { ENTITY_CREATE, ENTITY_DELETE, ENTITY_UPDATE } = types;
1557
1558 export const STATE_KEY = 'serviceCalls';
1559
1560 export const initialState = {
1561   allIds: [],
1562   byId: {},
1563 };
1564
1565 export default produce((draft, { type, payload }) => {
1566   switch (type) {
1567     case ENTITY_CREATE: {
1568       const { entity } = payload;
1569       const { id: entityId } = entity;
1570
1571       draft.allIds.unshift(entityId);
1572       draft.byId[entityId] = entity;
1573       break;
1574     }
1575     case ENTITY_DELETE: {
1576       const { entity } = payload;
1577       const { id: entityId } = entity;
1578
1579       draft.allIds.splice(
1580         draft.allIds.findIndex(svcId => svcId === entityId),
1581         1
1582       );
1583       delete draft.byId[entityId];
1584       break;
1585     }
1586     case ENTITY_UPDATE: {
1587       const { entity } = payload;
1588       const { id: entityId } = entity;
1589       Object.entries(entity).forEach(([key, value]) => {
1590         draft.byId[entityId][key] = value;
1591       });
1592       break;
1593     }
1594     // no default
```

```
1595   }
1596 }, initialState);
1597
1598 /**
1599  * NOME DO ARQUIVO
1600  * ./reducers/rtc.js
1601  */
1602 import immerReducer from 'src/utils/immer-reducer';
1603 import { types } from 'src/actions/rtc';
1604
1605 const {
1606   PEER_CONNECT,
1607   PEER_SET,
1608   STREAM_SET,
1609   SIGNAL_RECEIVE,
1610   SIGNAL_SEND,
1611 } = types;
1612
1613 export const STATE_KEY = 'rtc';
1614
1615 export const CONNECTION_STATE = {
1616   REQUESTED: 'REQUESTED',
1617   DISCONNECTED: 'DISCONNECTED',
1618   SIGNAL_SENT: 'SIGNAL_SENT',
1619   SIGNAL_RECEIVED: 'SIGNAL_RECEIVED',
1620   CONNECTED: 'CONNECTED',
1621   CONNECTED_STREAM: 'CONNECTED_STREAM',
1622 };
1623
1624 export const initialState = {
1625   connectionState: CONNECTION_STATE.DISCONNECTED,
1626   error: null,
1627   peer: null,
1628   localStream: null,
1629   remoteStream: null,
1630 };
1631
1632 const reducer = immerReducer(
1633   {
1634     [PEER_CONNECT]: (state, payload) => {
1635       state.localStream = payload.stream;
1636       state.connectionState = CONNECTION_STATE.REQUESTED;
1637     },
1638     [PEER_SET]: (state, payload) => {
1639       state.peer = payload.peer;
1640       state.connectionState = CONNECTION_STATE.CONNECTED;
1641     },
```

```
1642      [SIGNAL_SEND]: state => {
1643        state.connectionState = CONNECTION_STATE.SIGNAL_SENT;
1644      },
1645      [SIGNAL_RECEIVE]: state => {
1646        state.connectionState = CONNECTION_STATE.SIGNAL_RECEIVED;
1647      },
1648      [STREAM_SET]: (state, payload) => {
1649        state.remoteStream = payload.stream;
1650        state.connectionState = CONNECTION_STATE.CONNECTED_STREAM;
1651      },
1652    },
1653    initialState
1654  );
1655
1656  export default reducer;
1657
1658  /**
1659   * NOME DO ARQUIVO
1660   * ./selectors/service-call.js
1661   */
1662  /* eslint-disable */
1663  export const getActiveServiceCall = state => {
1664    const serviceCallId = state.ui.activeServiceCall;
1665    const serviceCall = state.entities.serviceCalls.byId[
         serviceCallId];
1666
1667    return serviceCall;
1668  };
1669
1670  /**
1671   * NOME DO ARQUIVO
1672   * ./store/configureStore.js
1673   */
1674  import { applyMiddleware, compose, createStore } from 'redux';
1675  import { apiMiddleware } from 'redux-api-middleware';
1676  import logger from 'redux-logger';
1677  import thunk from 'redux-thunk';
1678  import { devToolsEnhancer } from 'redux-devtools-extension'; //
       eslint-disable-line
1679  import io from 'socket.io-client';
1680
1681  import createSocketMiddleware from 'src/utils/socket-middleware';
1682  import rtcMiddleware from 'src/utils/rtc-middleware';
1683  import rootReducer from 'src/reducers';
1684
1685  const eventsRoot = process.env.REACT_APP_EVENTS_ROOT || '';
1686
```

```
1687 const configureStore = preloadedState => {
1688   const socket = io('/caller', {
1689     path: eventsRoot,
1690     transports: ['websocket'],
1691     autoConnect: true,
1692     reconnection: false,
1693   });
1694   const socketMiddleware = createSocketMiddleware(socket);
1695   const middlewares = [
1696     thunk.withExtraArgument({ socket }),
1697     apiMiddleware,
1698     socketMiddleware,
1699     rtcMiddleware,
1700     logger,
1701   ];
1702   const middlewareEnhancer = applyMiddleware(...middlewares);
1703
1704   const enhancers = [middlewareEnhancer, devToolsEnhancer()];
1705   const composedEnhancers = compose(...enhancers);
1706
1707   const store = createStore(rootReducer, preloadedState,
         composedEnhancers);
1708
1709   if (process.env.NODE_ENV !== 'production' && module.hot) {
1710     module.hot.accept('../reducers', () => store.replaceReducer(
           rootReducer));
1711   }
1712
1713   return store;
1714 };
1715
1716 export default configureStore;
```

Listing 7.3 – Projeto SAC Caller

8 APÊNDICE B - ARTIGO

Sistema de Atendimento ao Consumidor utilizando a tecnologia WebRTC

Lucas Rinaldi

[1]INE – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brazil

Abstract. *The goal of this paper is to show how to improve the way e-commerce, technical support and online services provide customer support. By building a communication system through video that can easily be included in companies' websites. Realtime video communication will be done using the WebRTC technology, which is relatively new but is expanding in a fast pace in video streaming. One if its advantages is that it makes unnecessary to use plugins or external libraries to transmit audio and video inside the browser, simplifying the customer support.*

Resumo. *O objetivo desse artigo é mostrar a forma como e-commerce, suporte técnico e serviços online podem ser melhorados. Construindo um sistema de atendimento ao consumidor com transmissão por vídeo que possa ser incluído em websites de forma simples e rápida. Para a transmissão de vídeos em tempo real será utilizada a tecnologia WebRTC, que apesar de ser relativamente nova, está expandindo horizontes na questão de transmissão de vídeos. Essa tecnologia torna desnecessário o uso de plugins e bibliotecas externas para comunicação através de áudio e vídeo no browser, simplificando o suporte técnico.*

1. Introdução

Com a crescente competitividade e concorrência de mercado, uma das maneiras de um negócio se destacar e conseguir sobreviver nessa realidade é investir no relacionamento com o cliente. Esse investimento busca melhorar o grau de satisfação do cliente, que possui um papel muito importante na longevidade do negócio.

Podemos identificar alguns problemas que afetam empresas no mercado atual em relação a este assunto:

- Empresas estão falhando em conhecer com precisão o perfil de seus consumidores;
- Há grande preocupação em adquirir novos clientes, mas não em retê-los. De acordo com [Farris et al. 2015], o preço para conseguir um novo cliente pode ser até 14 vezes maior que nutrir e manter um que já está na base;
- A tecnologia no atendimento ao cliente resultou em impessoalidade, com cada vez mais robôs atendentes no lugar de pessoas.

Observa-se a existência de diversas aplicações *web* voltadas para atendimento ao consumidor, com foco em soluções de *tickets*, análise e categorização dos mesmos.

Entretanto a correlação dos dois segmentos é pouco explorada, tanto no âmbito da Web, como com programas *desktop*, ou seja, sem a necessidade de instalação de programas e *plugins*. Para o usuário, a facilidade de acesso e a independência do sistema operacional são vantagens de uma aplicação web em relação à sua versão para uma plataforma específica.

Essa foi a principal motivação para o projeto: o desenvolvimento de uma aplicação web que faça a gerência de chamadas em vídeo e seja de fácil acesso para qualquer tipo de dispositivo. As chamadas iniciam-se por clientes que necessitam de assistência para o produto oferecido pela empresa, e são respondidas por atendentes designados para esse tipo de trabalho.

O modelo de atendimento por vídeo faz as empresas modernas retomarem a proximidade com os clientes. A tecnologia pode ser utilizada para sermos mais pessoais e alcançarmos uma gama maior de perfis de consumidor, por exemplo: o tipo de consumidor que está acostumado a interagir com o vídeo - meio de comunicação que se tornou uma das principais ferramentas de ajuda e comunicação na nossa sociedade atual.

De acordo com [Chazal 2015], o vídeo é o canal de crescimento mais rápido. O número de chamadas de vídeo subiu de 600 milhões em 2010 para 30 bilhões em 2015. Enquanto isso, 2 bilhões de minutos de chamadas via Skype são feitos todos os dias e 36% dos consumidores já expressam o desejo de terem suporte por meio de vídeo.

No entanto, a transmissão por vídeo é uma tecnologia relativamente nova em sistemas de atendimento, e somente 0,2% das empresas prestam atendimento via vídeo (de acordo com a [NewVoiceMedia 2014]). Empresas de consultoria, da área da saúde, *call* centers e advogados, por exemplo, já adotaram esse modelo.

2. Objetivo

2.1. Objetivo Geral

Desenvolver um Sistema de Atendimento ao Consumidor que funcione através de transmissão de vídeo. O sistema deverá dispor de uma área administrativa, onde um profissional responsável por prestar atendimento gerenciará seus chamados.

2.2. Objetivos Específicos

Os seguintes objetivos específicos são almejados por este trabalho:

- Estudar as soluções existentes para atendimento ao consumidor por chamadas de vídeo e identificar os problemas existentes;
- Desenvolver controle de acesso por parte dos atendentes.
- Desenvolver uma API baseada no modelo *RESTful* para cadastro de usuários e chamadas através de um servidor HTTP.
- Desenvolver um módulo que oferece conexões através de *WebSockets* para:
  - Receber requisições de chamadas em vídeo *realtime*.
  - Realizar o *handshake* necessário para conexão *peer-to-peer* entre navegadores.
- Implementar interface de aplicação web que suporte atualização em tempo real de componentes sem que seja necessário a atualização da página.

3. Projeto

Nessa secão apresentamos a estrutura, arquitetura e funcionalidades do projeto realizado no trabalho. Em primeiro momento foi feita uma análise dos requisitos necessários, para que então pudéssemos organizar todas as funcionalidades dentro do protótipo da aplicação e por fim projetar a arquitetura do *software* e a modelagem do banco de dados.

3.1. Análise de Requisitos

Dois tipos de requisitos foram levantados para a implementação do projeto: Requisitos Funcionais (RF), que são aqueles que definem as funcionalidades e como o *software* se comporta; e os Requisitos Não-Funcionais (RNF), que são responsáveis por especificar quesitos mais técnicos do *software*, como tecnologias, segurança e desempenho.

Para definição dos RF, foram analisados programas no estado da arte das duas categorias abordadas (videoconferência e atendimento ao consumidor) e coletadas funcionalidades consideradas importantes. Os RNF foram definidos a partir de estudos de aplicações Web modernas, com exceção do WebRTC que foi definido desde o principio. Requisitos funcionais serão apresentados com um diagrama de casos de uso, sendo cada caso um requisito.

Os RNF levantados são os seguintes (os itens 11, 12 e 13 foram baseados no trabalho [Fosser and Nedberg 2018, Sec. 4.3.1]):

- RNF 01: Utilizar a linguagem JavaScript.
- RNF 02: Arquitetura REST para API.
- RNF 03: Arquitetura com WebSockets para estabelecer a conexão P2P.
- RNF 04: Utilizar plataforma NodeJS para o servidor.
- RNF 05: Utilizar *framework* ExpressJS para API.
- RNF 06: Utilizar *framework* SocketIO para suporte a WebSocket.
- RNF 07: Utilizar *framework* ReactJS para renderizar interface no navegador.
- RNF 08: Utilizar PostgreSQL como gerenciador de banco de dados.
- RNF 09: Suporte a dispositivos móveis.
- RNF 10: Utilização de WebRTC para videoconferência.
- RNF 11: A taxa de transmissão deve estar acima de 1.0 Mbps.
- RNF 12: Porcentagem de perda de pacotes não deve ultrapassar o máximo de 10%.
- RNF 13: Limite da latência da comunicação de 1000 milisegundos.

Os RF levantados são representados como casos de uso na Figura 1.

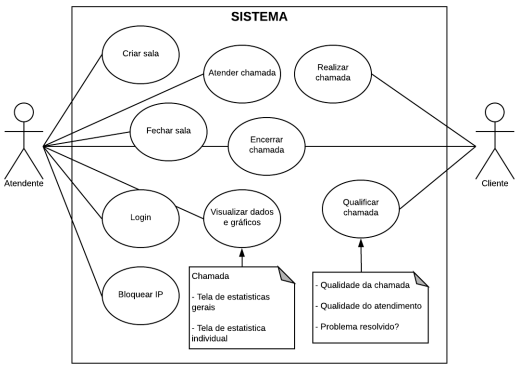

Figure 1. Requisitos Funcionais do sistema representados por casos de uso

Observamos na Figura 2 uma visão geral da arquitetura do sistema. Notamos que o modelo utilizado é o clássico cliente-servidor.

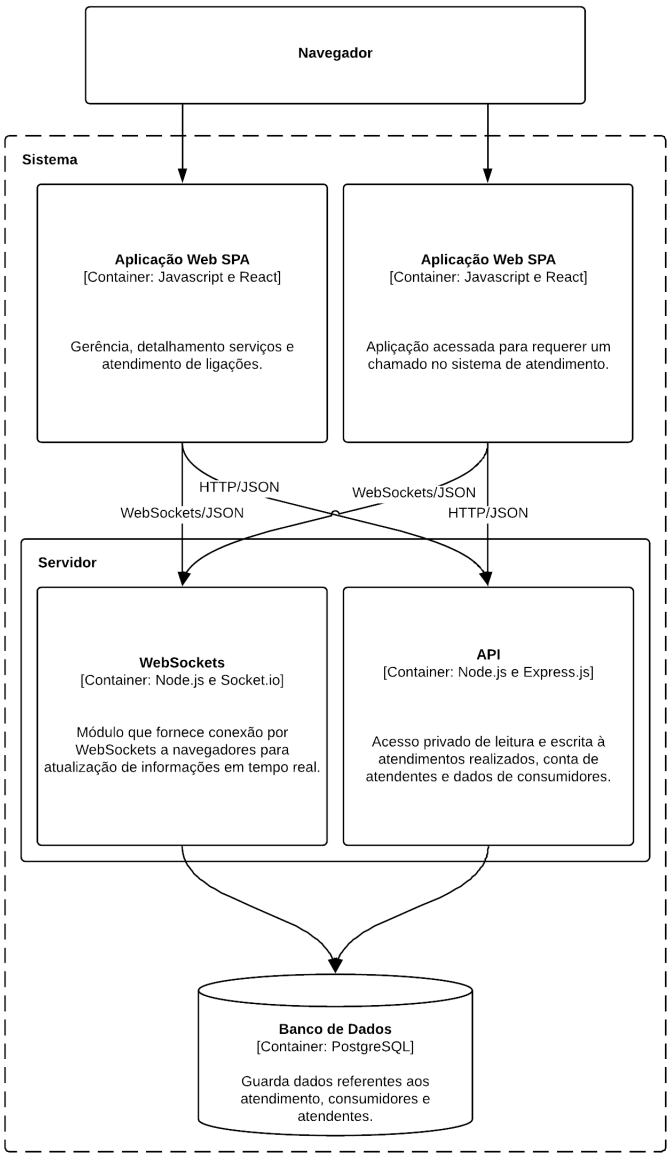

Figure 2. Arquitetura do sistema

3.1.1. Servidor

Todo o *back-end* foi implementado sobre a plataforma NodeJS. Essa tornou-se popular dentre os desenvolvedores nos últimos tempos devido à capacidade de executar código JavaScript fora do navegador.

Podemos elencar três componentes mais importantes no lado do servidor. O primeiro deles é a base, uma interface de programação de aplicação (API) nos modelos da arquitetura REST, implementada com o *framework* ExpressJS, utilizado para facilitar a construção de aplicações Web. O servidor da API, além de ser utilizado para salvar informações no banco de dados, também é responsável por iniciar a conexão através de *sockets*.

O segundo componente é outro *framework*, SocketIO, desenvolvido com o o objetivo de facilitar o *upgrade* de requisições HTTP e estabelecer a comunicação através de WebSockets. Esse *framework* abstrai as APIs complexas do navegador em padrões já conhecidos, fornecendo ao desenvolvedor uma outra arquitetura mais simples de usar.

Por último, temos o gerenciador de banco de dados PostgreSQL, que armazena os dados das chamadas, atendentes, administradores e clientes do sistema.

3.1.2. Cliente

O lado do cliente será sempre executado em um navegador, ou seja, foi inteiramente implementado com HTML e JavaScript.

Utiliza o *framework* React para renderizar interfaces. É baseado em composição de componentes, sendo cada um deles uma peça da interface. Esses componentes têm acesso às APIs nativas do navegador, podendo assim realizar requisições HTTP através de interfaces como AJAX.

É responsabilidade do cliente solicitar uma conexão *peer-to-peer* com um navegador remoto, conforme ilustrado na figura 3.

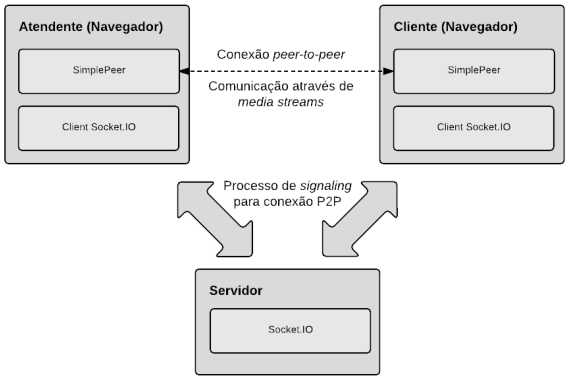

Figure 3. Conexão peer-to-peer através de WebSockets

Com a conexão estabelecida entre os dois pontos, a aplicação renderiza automaticamente uma nova tela, na qual o cliente consegue enxergar o atendente e vice-versa.

3.2. Modelagem do banco de dados

O diagrama de entidades, representado na figura 4, contém a modelagem do banco de dados da aplicação.

A entidade *User* representa os atendentes do serviço de atendimento ao consumidor. Todo atendente necessita fazer login no sistema para poder criar salas e atender clientes. Esse mesmo usuário tem permissão para visualizar dados e estatísticas de chamadas.

Para representar os clientes que realizam chamadas de suporte criamos a entidade *Customer*. Essa entidade não necessita de login para solicitar um atendente, visto que o objetivo do sistema é prover uma maneira fácil para pessoas obterem resolução de problemas.

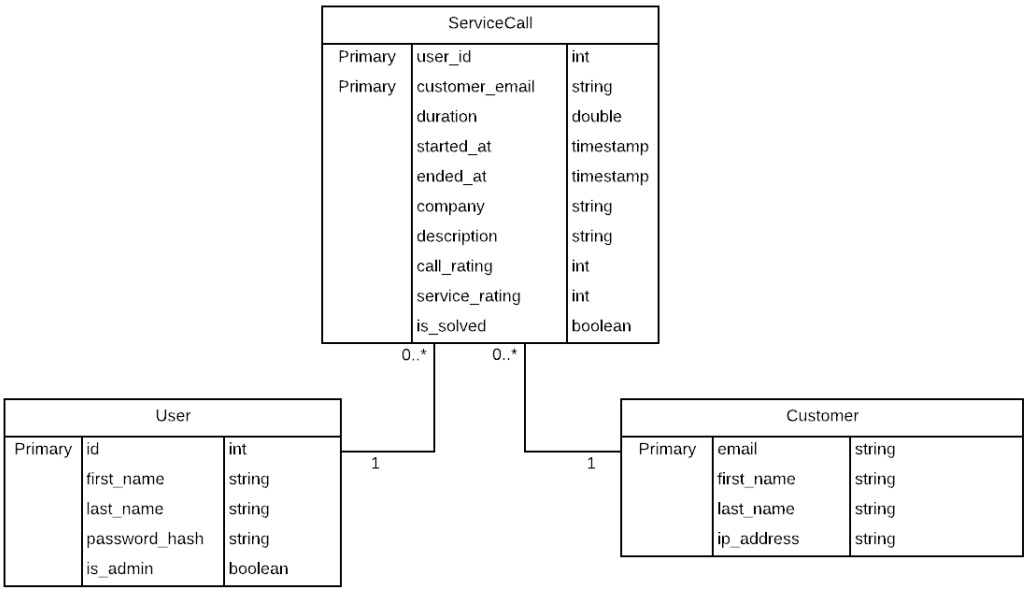

Figure 4. Entidades do banco de dados

Por fim, as chamadas de suporte são representadas pela entidade *ServiceCall*. Essa entidade possuí o tempo de duração de atendimento e faz a relação entre atendente e consumidor. Foram adicionados também dados da chamada, como avaliação e se o problema foi resolvido ou não, para que seja possível registrar estatísticas e criar uma página com informações para os atendentes.

4. Implementação

Com base na modelagem de classes e nos casos de uso levantados foi criada uma arquitetura para suprir as necessidades do sistema e do problema apresentado no trabalho.

4.1. Visão geral do sistema

O sistema de atendimento ao consumidor foi pensado de maneira a fornecer ao usuário uma facilidade ao se conectar com seu próprio cliente. A decisão de dividir o projeto em múltiplas partes veio da necessidade do *sac-caller* ser usado em dispositivos móveis, ou seja, era necessário uma aplicação pequena, que não consumisse muitos dados do pacote de internet do usuário para baixá-la.

Observamos na Figura 5 como os componentes do sistema estão organizados. O projeto *SAC API* é um servidor com uma API REST que possui suporte a conexão *WebSockets* para a facilidade de troca de mensagens. Já SAC *Manager*, *Caller* e *Form* são os três constituídos somente de HTML, JavaScript e CSS, sendo os dois primeiros aplicações feitas em ReactJS.

Na Figura 6 está explicado como é o fluxo para realizar uma conexão entre cliente e usuário:

5. Considerações finais

O presente trabalho teve como objetivo resolver o problema da impessoalidade nos serviços de atendimento devido ao aumento da tecnologia nos processos, como podemos

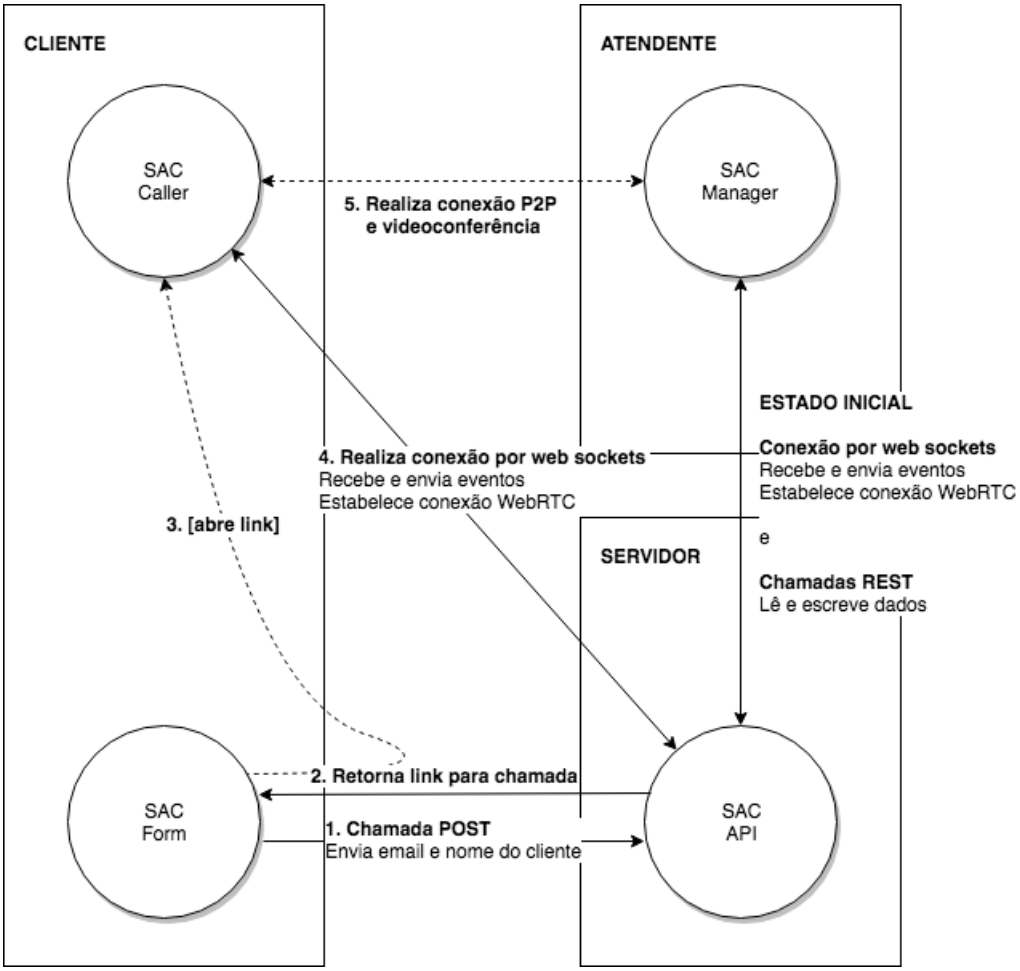

Figure 5. Visão geral dos projetos do sistema

ver na seção 1.

Com o aumento de serviços disponibilizados *online* a necessidade de lojas físicas é cada vez menor, surge a seguinte dúvida: como tornar o atendimento ao consumidor mais pessoal. Essa resposta o trabalho se propôs a resolver através das chamadas de vídeo, integrando as mesmas em um sistema de atendimento ao consumidor.

Após definida a solução (conversa em vídeo) para o primeiro problema, foram elencadas algumas tecnologias que serviriam para realizar o *software*, isso levando em conta requisitos para que o atendimento não fosse muito trabalhoso para nenhuma das partes. Ambos objetivos foram alcançados e podem ser vistos na Seção **??** do projeto.

Realizar esse trabalho proporcionou um aprendizado importante sobre como enxergar e arquitetar a partir de uma visão geral até detalhes de implementação. Foi preciso estar em contato com aplicações de gerência de informações, como banco de dados. Descobrir uma maneira de distribuir essas informações em diferentes aplicações, percorrendo áreas como protocolos de rede, conexão entre processos e arquitetura REST. E por último, foi crucial para a aplicação o conceito de transmissão de informações em tempo real, realizada com *web sockets*.

Uma aplicação com diversas partes conectadas (cliente, empresa, atendente) en-

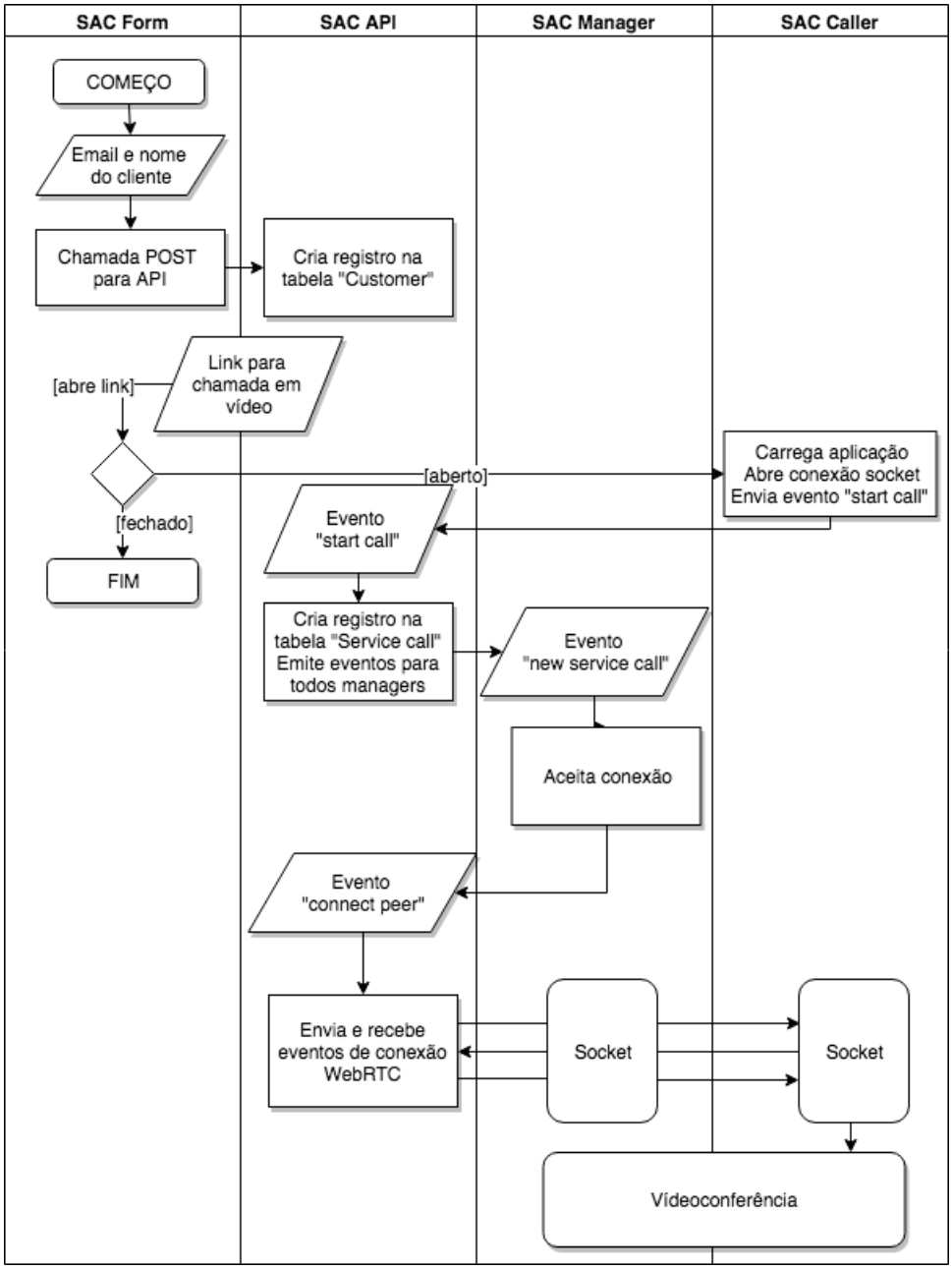

Figure 6. Fluxograma conexão WebRTC entre cliente e usuário

globa diferentes áreas de conhecimento, em virtude disso pode-se dizer que o projeto foi multi-disciplinar.

5.1. Trabalhos futuros

Existem algumas extensões do sistema que podem ser adicionadas para melhorar o resultado da ferramenta com as empresas e os seus consumidores.

Um dos motivos para incluir um formulário de *feedback* no projeto é a coleta de dados dos atendimentos. As informações entradas atualmente (qualidade do serviço e chamada) não vão muito a fundo na questão do problema, porém no futuro podem ser

incluídas outras, como por exemplo: categorização do problema e do cliente; palavras-chave e descrição; entre outras.

Com esses dados um banco de informações pode ser realizado acompanhado de um *helpcenter* que serviria para o auto-atendimento de clientes e solução de problemas simples. Um outro uso seria a análise desses dados para descobrir informações importantes sobre o produto, seu meio de produção e possíveis falhas que possam existir no processo.

O principal foco da aplicação foi realizar a chamada de vídeo através de uma conexão *peer-to-peer* e conseguir gerenciar esse fluxo chamada-resposta com diversos atendentes, portanto melhorias de design e performance são uma ótima escolha:

- Testes de usabilidade seriam um próximo passo para analisar necessidades tanto das empresas e atendentes como dos clientes; e
- Testes de performance, de conectividade e de qualidade de chamada também seriam úteis para avaliar o funcionamento a aplicação, visto que ela não foi testada em ambiente controlado com simulação de múltiplas conexões simultâneas.

References

Chazal, A. (2015). The untapped potential of ecommerce video chat. Online.

Farris, P., Bendle, N., Pfeifer, P., and Reibstein, D. (2015). *Marketing Metrics: The Manager's Guide to Measuring Marketing Performance*. Pearson FT Press; 3 edition (September 6, 2015).

Fosser, E. and Nedberg, L. (2018). Quality of experience of webrtc based video communication.

NewVoiceMedia (2014). What contact centres are doing right now. Technical report, NewVoiceMedia.